Trajectory Mining in the Context of the Internet of Things

Xiaoting Wang

Submitted in total fulfilment of the requirements of the degree of

Doctor of Philosophy

Department of Computing and Information Systems
THE UNIVERSITY OF MELBOURNE

June 2017

Abstract

THE Internet of Things (IoT) is a technological revolution that is rapidly reshaping our society with ubiquitous sensing devices. As more sensors are being deployed, the amount of data collected is also growing significantly, leading to the increasingly important role of data mining in the IoT.

One specific type of data frequently captured by sensors in the context of the IoT is *trajectory data*, which has attracted considerable attention lately. Various types of trajectory data have become available, such as user check-ins, vehicle GPS traces and human activity profiles recorded by smart phones. As trajectories can be generated by a large variety of sensors, many challenges arise in different application scenarios in the IoT that are associated with trajectory data mining.

In this thesis, we address three trajectory data mining challenges in three different application scenarios of the IoT, namely, community, transport and healthcare applications. The first challenge is in social and community applications. The check-ins and geo-tagged photos submitted by a user on a social networking site can be regarded as trajectories that reflect the popularity of the points-of-interest (POI) being visited and the personal preference of the user. This data can be used to recommend trips to a tourist. However, the challenge of trip recommendation not only lies in searching for relevant POIs to form a personalized trip, but also selecting the best time of day to visit the POIs. Popular POIs can be too crowded during peak times, resulting in long queues and delays. To improve the quality of the solution of automated trip recommendation, we propose the Personalized Crowd-aware Trip Recommendation (PersCT) algorithm to recommend personalized trips that also avoid the most crowded times of the POIs using trajectories and pedestrian sensing data. Our results on a real-life dataset show that it is possible to achieve a balance between conflicting objectives in trip recommendation, such as satisfying user interests while reducing the crowdedness of the trips.

The second challenge is in transport applications. Despite the recent introduction of advanced technologies such as Intelligent Transportation Systems (ITS), traffic congestion remains a major challenge to urban designers and transport authorities. Current ITS technologies usually employ induction loop sensors or street cameras to monitor the traffic conditions. However, monitoring the change in the traffic flows as a result of external events can be difficult. Therefore, we propose a framework to analyse changes in traffic flows due to road closure events based on GPS trajectory data. We employ ideas from contrast mining and frequent itemset mining to define, characterize and visualize the changes. The effectiveness and robustness of this framework are shown by three experiments using real taxi trajectories as well as traffic simulations in two different cities.

The third challenge is in healthcare applications. The increasing health care cost has motivated the development of remote health monitoring. Home-based rehabilitation from conditions such as stroke is usually unmonitored and the progress of recovery is difficult to assess. Typically, a patient routinely revisits the hospital to seek feedback on the rehabilitation progress, which is costly and inefficient. The use of wearable sensors to capture limb movement information can be a cost-efficient alternative to frequent hospital revisits. In this context, one important problem of interest is to recognize specific actions performed by the person in daily life so that a physician can easily observe the patient's movement from the continuous data stream sent by the sensors. We propose the use of a 3D trajectory reconstruction algorithm to process the raw sensor data and extract trajectory features that can improve action recognition accuracy. We also propose a clustering-based classifier for use with the trajectory features, and show that our proposed approach outperforms several benchmark classifiers in recognition accuracy.

We conclude this thesis by presenting a number of future research directions that may be promising for trajectory mining in the IoT.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Xiaoting Wang, June 2017

Acknowledgements

This thesis would not have been possible without the continuing support, mentorship and encouragement from my supervisors. I would like to thank Prof. Christopher Leckie and Dr. Tharshan Vaithianathan for their inspiring advice, dedication and academic and psychological assistance throughout the degree. This has been a difficult but rewarding journey and I am really grateful that I have had them as my supervisors. I would also like to thank my advisory committee chair Prof. Timothy Baldwin for his guidance and support.

I would like to express my gratitude to Dr. Jeffery Chan who offered helpful advice, collaboration and mentoring during my degree. I would like to extend my thanks to Dr. Hairuo Xie and Dr. Sofia Suvorova for their collaboration and helpful discussions. I would also like to thank Prof. Michael Houle for his support and guidance during my internship at NII. I would like to offer special thanks to my friend Mr. Kwan Hui Lim for his collaboration and the sharing of his data.

I would like to thank NICTA and Data61, which generously provided a full PhD scholarship and several travelling grants throughout my degree. I would also like to express my appreciation for the research facilities provided by the University of Melbourne.

I would like to show my deepest gratitude to my parents who have always been supportive during my study, both financially and psychologically. I am deeply indebted to them and I will always thank them for being there for me.

Preface

The following is the list of publications that have arisen from this thesis, and the corresponding chapters that contain material from these publications.

Paper P1: Chapter 3

- Wang, X., Leckie, C., Chan, J., Lim, K.H., Vaithianathan, T. (2016). Improving Personalized Trip Recommendation by Avoiding Crowds. In the 25th ACM International Conference on Information and Knowledge Management (CIKM 2016). ACM.

Paper P2: Chapter 4

- Wang, X., Leckie, C., Xie, H., Vaithianathan, T. (2015). Discovering the Impact of Urban Traffic Interventions Using Contrast Mining on Vehicle Trajectory Data. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2015), (pp. 486-497). Springer International Publishing.

Paper P3: Chapter 5

- Wang, X., Suvorova, S., Vaithianathan, T., Leckie, C. (2014). Using trajectory features for upper limb action recognition. In IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2014), (pp. 1-6). IEEE.

To my parents

Contents

List of Figures

List of Tables

List of Abbreviations

ACO	Ant Colony Optimisation
CF	Collaborative Filtering
E2ES	Emerging 2-Edgeset
EKF	Extended Kalman Filter
EnES	Emerging n-Edgeset
EP	Emerging Pattern
FEN	Frequent Emerging Network
HMM	Hidden Markov Models
IBCF	Item-based Collaborative Filtering
IMU	Inertial Measurement Unit
IoT	Internet of Things
LOF	Local Outlier Factor
LBSN	Location-Based Social Network
MF	Matrix Factorisation
nES	n-Edgeset
NoE	Number of Edgesets
OP	Orienteering Problem
PCA	Principal Components Analysis
POI	Point-of-Interest
SVM	Support Vector Machine
UBCF	User-based Collaborative Filtering

Chapter 1

Introduction

THE recent boom in the wireless Internet has led to a rapid increase in the number of connected devices on the Internet. In the past, a majority of devices on the Internet were associated with a single user, i.e., the Internet of people. Today, more and more sensors are connected to the Internet, which form a key part of the Internet of Things (IoT), and are facilitating the development of smart cities and smart homes. For example, home appliances that are connected to the Internet can be controlled remotely by a user via a smart phone [104]. Taxis equipped with a GPS sensor can report their location in real-time to allow smart allocation of passengers [150]. A network of connected rubbish bins in a city can be used to monitor the fill-level of the bins and allow more efficient collection routes to be designed [50]. Such applications were not possible in the past without this type of sensing infrastructure.

As more devices are connected to the Internet and more sensors are being deployed, the amount of data that is being collected by the sensors is also increasing significantly. The analysis and "mining" of such data will be a key focus to effectively utilise the IoT infrastructure. In this context, a specific type of data called *trajectory data* has attracted much attention. Trajectory data are the time series captured by sensors that are observing moving objects. Various types of sensors and objects can generate trajectory data, such as vehicle GPS traces, pedestrian patterns captured by street cameras or sensors, check-in behaviour on a social networking site, and human activity profiles recorded by smart phones, smart watches or wearable sensors. Data mining for such a large variety of data sources is not a trivial task due to the dynamic environments and the challenges in merging data with different quality and granularity.

In this thesis, we focus on the challenge of data mining for trajectory data in the context of the Internet of Things. Specifically, we identify three types of trajectory data that are commonly

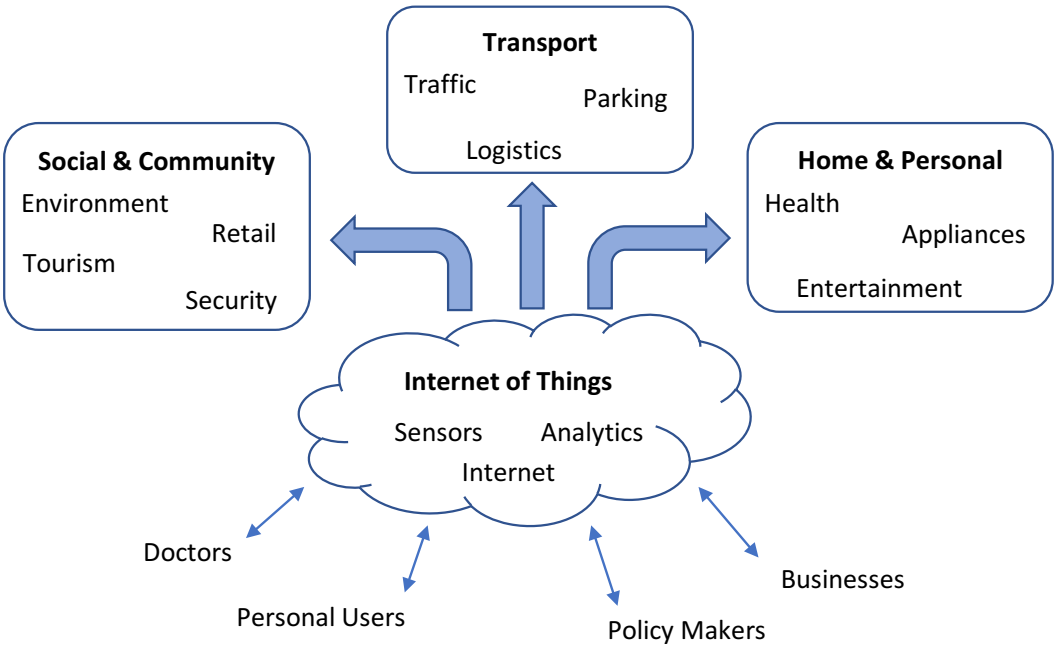


Figure 1.1: Framework of the Internet of Things

found in this context, user trajectories, vehicle trajectories and movement trajectories of human body parts. Each of the three types of trajectories can be useful in one application of the IoT: user trajectories → community, vehicle trajectories → transport, and movement trajectories of human body parts → healthcare. These three application scenarios cover a large social spectrum and will have a substantial impact on society as a whole. In each of these application scenarios, we identify an individual problem that can benefit from the use of trajectory data. Before going into the details, we start by discussing the motivation of this study.

1.1 Motivation

The Internet of Things has been named as one of the six "Disruptive Civil Technologies" by the US National Intelligence Council (NIC) [9]. NIC predicts that "by 2025 Internet nodes may reside in everyday things: food packages, furniture, paper documents, and more" [9]. This ubiquitous sensor network will be the enabling technology for many novel applications. A schematic of the IoT framework is illustrated in Figure 1.1. Multiple parties, including ordinary users, policy makers, doctors and industries will be connected to the IoT infrastructure [60]. Central to this

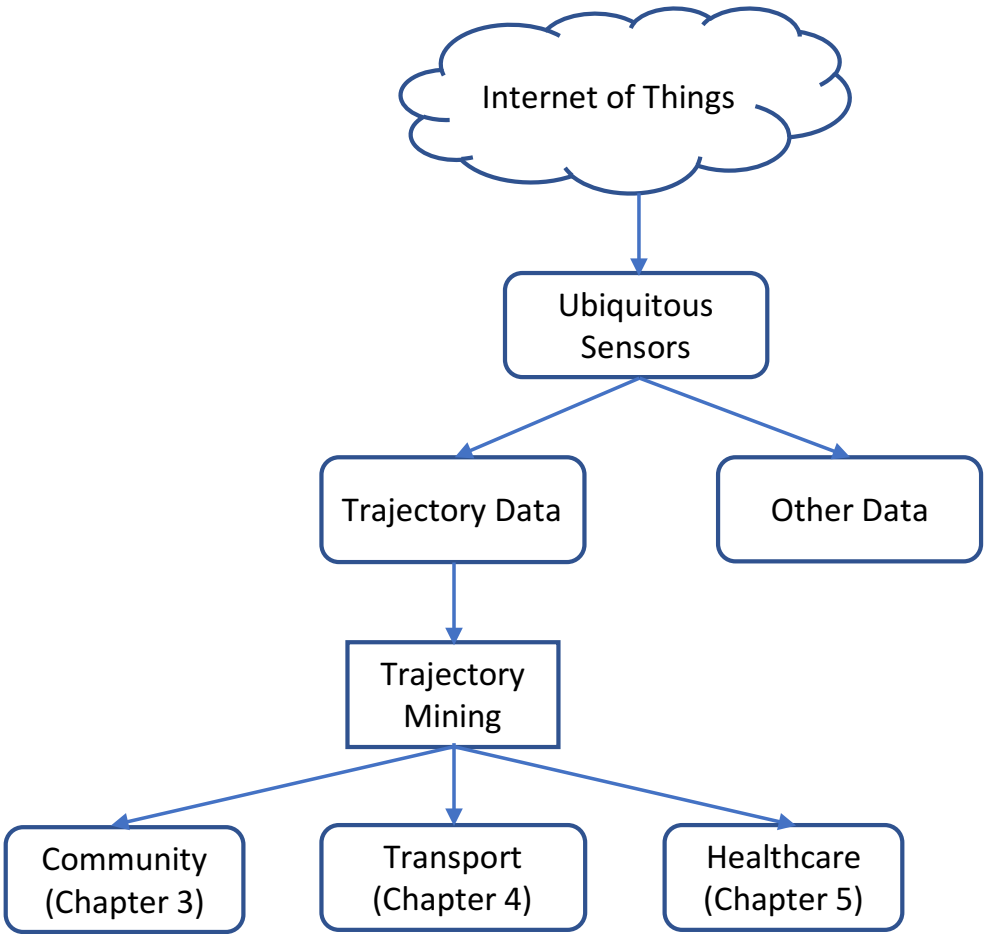


Figure 1.2: Organisation of this thesis

framework are the smart sensing and analytics tools that facilitate knowledge discovery and data analysis in various scenarios to provide end users with useful information and insights for decision making.

By connecting objects and devices to form a ubiquitous network, the information gathered can be used to address many challenging problems. Gubbi et al. [60] outline four potential application scenarios in Figure 1.1. In this thesis, we focus on mining a specific type of data, spatio-temporal trajectories, for three application scenarios: community, transport and healthcare (Figure 1.2). We address one existing challenge in each of these scenarios and propose solutions that improve on the current data analysis techniques in each application scenario.

Specifically, in community applications, one of the existing challenges is decision making for individuals based on crowd information. Our context for this problem is in trip and tourist

destination recommendation. During holiday seasons or special events, it is often difficult for an ordinary user to be informed about the overall crowdedness of popular attractions. Users may predominantly favour the most popular location, which may lead to over-crowdedness and even fatal accidents [59]. For trip recommendation services, it is desirable to inform tourists about the crowdedness of attractions and make recommendations based on such information. Currently, this is a difficult problem, mainly due to the lack of crowd data in most cities. However, when such data does become available, it is possible to recommend trips that take into consideration both crowdedness and the personal interests of users by combining crowd data with user visit trajectories. Chapter 3 presents such a framework, which recommends trips that satisfy user interests and are less crowded.

In transport applications, congestion reduction is a vital problem of interest to both policy makers and ordinary motorists. Besides reducing traffic by investing in public transport, carpooling or a congestion tax, it is believed that providing sufficient route information and driving guidance can reduce travel time [8]. In the case of a road closure event or traffic accident, the change in the flow of traffic can be difficult to foresee. In these situations, drivers may be able to identify the road segment under direct impact of the event from radio reports. However, the increase of traffic flows in nearby roads as a result of the event is difficult to quantify and drivers can be ill-informed about the best detour route to take and the likely increase in travel time. In Chapter 4, we devise a solution that uses trajectory mining on vehicle GPS data to quantify the impact of road events.

In healthcare applications, we focus on the problem of action recognition with wearable sensors. Recent smart phones and smart watches are usually equipped with a large variety of sensors, such as accelerometers, gyroscopes, barometric pressure sensors and heart rate sensors. By connecting these devices to the Internet, physicians will be able to remotely monitor the recovery progress of patients who have suffered from conditions that require long-term home-based rehabilitation such as stroke. To facilitate interpretation of the data for the physicians, it is necessary to recognise the actions performed by the patient. For example, when a patient's arm mobility is affected by a stroke, physicians need to monitor the progress of the patient in performing a range of routine tasks [111]. This can be achieved by reconstructing arm motions of the patient into trajectories, and classifying the actions using the trajectory data. In Chapter 5, we present such an approach that uses trajectory mining to solve this action recognition problem.

1.2 Thesis Structure and Contributions

This thesis is organized as follows.

Chapter 2. In this chapter, we introduce background knowledge on trajectory data mining (TDM) and the Internet of Things (IoT). A high-level literature review on the current progress of research on TDM and IoT is given. The detailed review of work more related to our focus will be given in the following three contribution chapters.

Chapter 3. In this chapter, we study the problem of travel route optimization using user-generated trajectories. Algorithm-based personalized trip recommenders have been gaining attention as more user-generated trajectory data becomes available [87]. The challenge of trip recommendation not only lies in searching for relevant points-of-interest (POIs) to form a personalized trip, but also selecting the best time of day to visit the POIs. Popular POIs can be too crowded during peak times, resulting in long queues and delays. We propose the Personalized Crowd-aware Trip Recommendation (PersCT) algorithm to recommend personalized trips that also avoid the most crowded times of the POIs. We model the problem as an extension of the Orienteering Problem [56] with multiple constraints. We extract user interests by collaborative filtering and we propose an extension of the Ant Colony Optimisation algorithm [48] to merge user interests with POI popularity and crowdedness data to recommend trips. We evaluate our algorithm using foot traffic information obtained from a real-life pedestrian sensor dataset [2] and user travel histories extracted from a Flickr photo dataset [87]. Our major contributions are:

- We formulate the personalized crowd-aware tour recommendation problem.
- We combine multiple objectives and propose the PersCT algorithm that can efficiently find a solution that achieves a balance between conflicting objectives such as user interest and crowdedness of trips.
- We evaluated the effectiveness of our algorithm using a Flickr photo dataset and pedestrian count dataset, both in the city of Melbourne.

Publication

- Wang, X., Leckie, C., Chan, J., Lim, K.H., Vaithianathan, T. (2016). Improving Personalized Trip Recommendation by Avoiding Crowds. In the 25th ACM International Conference

on Information and Knowledge Management (CIKM 2016). ACM.

Chapter 4. In this chapter, we present a framework to discover contrast patterns in trajectory data, with a focus on road traffic analysis. Contrast patterns are subgroups of features (or items) whose frequency of occurrence varies significantly between two given datasets [46]. By identifying contrast patterns in trajectory data, changes in the grouping of trajectories can be discovered. In the case of road traffic analysis, studying contrast patterns can potentially enable road authorities and motorists to gain a better understanding of the impact of events, such as road closures, on the traffic. We compute contrast patterns by finding connected road segments with significantly different traffic levels before and after the intervention using a notion of *growth rate*. Frequent sub-networks of the overall traffic network are then discovered to reveal the regions that are most affected. We perform three experiments to evaluate the effectiveness and robustness of this framework using real taxi trajectories and traffic simulations. The results show that this method can discover interesting hidden relationships inside the traffic network. Our major contributions are:

- *Traffic Network Modelling.* We model the road traffic network of a city as a graph G, generate "n-Edgesets" and analyze vehicle trajectories to find the traffic volume on connected subsets of edges in G.
- *Mining Emerging n-Edgesets.* We propose the MineEmergingEdgeset algorithm to extract the n-Edgesets with significantly different traffic patterns before and after the traffic intervention (the Emerging n-Edgesets).
- *Mining Frequent Emerging Network.* We propose the MineFreqNetwork algorithm to find frequently occurring emerging networks using Emerging n-Edgesets. This frequent network reveals the most severely affected sub-network of the city as a result of the intervention.
- *Experiment with real-life and simulated datasets.* We perform three experiments to evaluate the effectiveness and robustness of this framework using real taxi trajectories and traffic simulations.

Publication

- Wang, X., Leckie, C., Xie, H., Vaithianathan, T. (2015). Discovering the Impact of Urban Traffic Interventions Using Contrast Mining on Vehicle Trajectory Data. In Pacific-Asia

Conference on Knowledge Discovery and Data Mining (PAKDD 2015), (pp. 486-497). Springer International Publishing.

Chapter 5. In this chapter, we study trajectory classification with a focus on action recognition using wearable motion sensors. Low-cost wearable sensors are increasingly being used to capture and analyze limb movement in home and health care applications [110]. However, classification of limb motion is still a difficult task due to large variations in sampled movement trajectories produced by the same subject. We presents an algorithm for the detection and classification of arm motion from data collected by wearable inertial sensors. High level arm trajectory features are obtained from raw sensor data using a sensor orientation tracking algorithm and an arm model. The features are then used in a clustering-based classifier. In the classifier training stage, features are clustered using the K-means algorithm [64], and a histogram of "key poses" is generated from the clustering as a template for each class. In the recognition stage, new data are segmented and matched to the templates. Experiments on human subjects show that by using trajectory features in the proposed approach, we can achieve higher accuracy than a range of benchmark non-temporal classifiers [28]. Our major contributions are:

- We use a sensor orientation tracking algorithm and an arm model to extract trajectory features, which is novel in action recognition.
- We propose the use of the K-means clustering algorithm in the training of a classifier to obtain histograms as templates of the actions.
- We match the unknown action data with the histograms using a nearest neighbour algorithm.
- We conducted experiments with human subjects to capture daily activities and evaluated the proposed method with several benchmark algorithms.

Publication

- Wang, X., Suvorova, S., Vaithianathan, T., Leckie, C. (2014). Using trajectory features for upper limb action recognition. In IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2014), (pp. 1-6). IEEE.

Chapter 6. We conclude by summarizing the results presented in this thesis. We then discuss several opportunities for further research based on the proposed approaches in this thesis and suggest a number of future research directions that may further advance trajectory data mining in the IoT.

Chapter 2

Background on Trajectory Data Mining in the IoT

In this chapter, we present the background to trajectory data mining in the IoT, and give an overview of some related work in this area. The detailed literature reviews will be included in the next three chapters, and this chapter aims to provide a high level introduction of the field.

2.1 Introduction

The first use of the term "Internet of Things" is attributed to the research in advanced sensing technologies associated with Radio-frequency IDentification (RFID) tags [9]. RFID tags are electromagnetic labels that can be easily attached to objects and read by a reader device. RFID tags can be passive, active or battery-assisted passive. For passive tags, a response containing its ID information is received when a reader sends a signal to the tag. This type of tag is frequently used in retail and supply chain applications [60]. Active tags are powered by on-board batteries, and its ID signal is periodically transmitted to the reader, which can be useful in logistic applications such as port cargo management where the operator constantly needs to know the location of each cargo box [72].

Although RFID systems have been used in a large number of applications [102], the information being transmitted is limited to the ID of the tag. As mobile computing and low power sensing technologies have matured in recent years, a large variety of sensors are beginning to be deployed as wireless sensor networks to collect, process and analyse more sophisticated data [60] in the IoT paradigm. One such data type is spatio-temporal trajectories. A trajectory is a sequence of time-stamped coordinates generated by an object or part of an object in 2 or 3-dimensional space. For example, we may view geo-tagged tweets or "check-ins" submitted by a user as the trajectory

of a user moving on the map: $Tr : L_1 \rightarrow L_2 \rightarrow ... \rightarrow L_M$, where each point L_i is a location with a time stamp $< lat, long, time >$. Some common sources of trajectory data as summarised by Zheng et al. [161] are:

1. location histories uploaded by users of social media [164]
2. location histories of vehicles reported by GPS sensors [140]
3. movement trajectories of animals [84]
4. motion trajectories of natural phenomena such as hurricanes and ocean currents.

The trajectory data in the above examples are all trajectories of whole objects. In IoT applications, trajectories may also arise from part of an object, such as the arms of a human equipped with wearable sensors [127], or robotic arms [45, 125]. Therefore, in this thesis we expand the definition of trajectory data to include trajectories of rigid bodies as well as the locations of objects as defined in Zheng et al. [161].

In the following sections, we introduce how trajectory mining is used in three application scenarios that are most related to our research, namely, in social/community, transport and healthcare applications.

2.2 Trajectory Mining for Social Applications

Research in trajectory mining for social networking began before the widespread adoption of smart phones and location-based social networking. Microsoft Research has been a pioneer in the field of social trajectory mining. One of the earliest works that involves capturing the daily routines of users is the SenseCam project [67], where each user wore a custom-designed camera unit. The unit can be configured to automatically take pictures of the scene in front of the user using a light sensor. Although no GPS location information was collected in this project at the time, it can be seen as one of the early attempts in capturing user visit histories. By analysing the content of the image, the location of the user can potentially be identified [52]. The disadvantage of identifying locations from image content is that the privacy of other users cannot be guaranteed and the volume of data stored can be large.

In 2007, Counts et al. [36] proposed SlamXR, a system that uses a prototype sensor board to capture GPS locations, acceleration, air pressure and temperature information from users. The system has three functionalities: travel route sharing, route visualisation and people search. Volunteers were recruited to carry the device around and log their daily movement trajectories. A web interface was built for the software front-end to visualise the trajectories. Features such as average speed and altitude were extracted from each trajectory, and a similarity score was evaluated between pairs of the trajectories. Routes with similar features receive a high similarity score, and therefore may be displayed to the users as suggested routes. The system also allows a user to search for desired trajectories and view the users who have submitted these trajectories, which can potentially be used for social networking. However, the system does not explicitly recommend similar users, and a user must browse the trajectory database to search for other users, which can be impractical if the trajectory database is large.

The GeoLife project [85, 99] was a major step forward in social trajectory mining. Trajectory data from 182 users in a period of over five years (April 2007 to August 2012) were collected using GPS loggers and GPS-phones. As the data are unlabelled trajectories, the researchers proposed a solution to the problem of *stay point detection*, which is to detect the point-of-interest (POI) where a user is currently staying. After detecting the POIs, the trajectories were clustered with a hierarchical graph. Similar users were found by looking for trajectory sequences that share similarity in the hierarchy structure. In addition to similar user discovery and recommendation, the researchers also recommended locations to users based on similarity with the visit history [85].

In more recent social trajectory mining literature, large datasets retrieved from social networking sites (SNS) (Twitter, Flickr, Foursquare, etc.) that contain user check-ins or geo-tagging information were used to recommend people, locations and trips [79, 87, 151, 162]. The use of SNS data has a major advantage: the data collection is significantly cheaper as users will upload trajectories voluntarily. Compared with the GeoLife Project [99], the number of users studied by these work are significantly larger. For example, the Gowalla dataset used in [151] contained 10,162 users whereas the GeoLife project collected data from only 182 users. The disadvantage is the loss in data granularity. Users tend to post to SNS sporadically, which results in sparse data. For example, two consecutive check-in events may be separated by a large time window. Even worse, users perhaps only submit a check-in at a new location (e.g., a new restaurant) or when

something interesting has occurred. The consequence of data sparsity is that it may be difficult to recommend the best time to visit each point-of-interest as the data collection may require a very large user base and an extended amount of time. The works that perform time-based location or trip recommendation [33, 151, 152] tend to recommend popular times of visit, when POIs can be in high demand and over-crowded. This motivates the use of external sensors to monitor the crowdedness of POIs to provide accurate information to visitors, especially for cities that offer mobile tour guide services. Although the cost of sensor installation may be higher than online data collection, only a modest number of sensors may be needed to cover most tourist attractions and popular destinations in a city. In Chapter 3, we present a trip recommendation framework that uses pedestrian sensors to monitor POI crowdedness, and then recommend trips so that the objective score based on the combination of the interest of the user and the crowdedness of the trip is optimised.

2.3 Trajectory Mining for Transport Applications

GPS-equipped vehicles have become increasingly common in everyday life. Taxis are usually installed with GPS devices so that the operating company can monitor the locations of the vehicles to offer services such as online booking. The data captured from these GPS sensors are vehicle trajectories, from which rich traffic information can be extracted. This information can be used in various transport applications such as resource allocation [150], travel time prediction [140] and identifying systematic problems in road infrastructure [91].

Before the introduction of real-life vehicle trajectories in data mining, data in the field of transport research are usually collected from two sources, traffic simulation [58] and inductive loop detectors placed on the streets [142]. Simulation is inexpensive and can be carried out on a large scale. However, the data may not accurately reflect the actual traffic conditions. Inductive loop detectors count vehicles by detecting the difference in the current generated in a wire loop when vehicles with metal frames pass over the loop, and is therefore a highly accurate method to measure the traffic volume. However, the cost of installing loop sensors is high and sensor coverage is often only limited to major roads. Moreover, inductive loop detectors do not provide detailed information at the level of the trajectories of individual vehicles. Consequently, there has

been growing interest in the analysis of vehicle trajectories based on GPS data.

The T-Drive project [98, 147] was one of the pioneering projects in trajectory mining with vehicle GPS data. The purpose of the project was to give smart driving directions to reduce travel time by learning from experienced taxi drivers. GPS trajectory data was collected from 30,000 taxis in Beijing over a period of 3 months. The authors identified these challenges for the problem of smart navigation using trajectories: (1) data sparsity of the trajectories, and (2) uncertainty in GPS samples. These challenges commonly occur in other vehicle trajectory mining problems and will be elaborated next.

Data sparsity of GPS signals is a challenge caused by the sampling rate of GPS devices. Many on-board vehicle GPS devices report the location every one to three minutes in order to save power [148]. This is sufficient for approximate tracking purposes as in online booking. However, if the GPS samples are distributed across the whole city, each road segment tends to receive a limited number of samples. Therefore, the temporal density of the GPS data is relatively low. The authors of [149] countered this problem by aggregating vehicle volume counts into hourly time slots so that sufficient data are available in each time slot. In contrast, spatial aggregation was investigated by Liu et al. [91]. Instead of calculating the vehicle count per-road, Liu et al. computed vehicle counts for larger areas on the map using a raster-based segmentation algorithm [91], and thus more data were available per time slot. Liu et al. [91] could thus split the time slots using 15-minute intervals, as compared to one-hour intervals used by Yuan et al. [149].

As GPS signals may suffer from noise and drift issues [112], the coordinates obtained from the sensors directly may not exactly follow the actual path traversed by the vehicle. In the geo-spatial research community, this is referred to as the *map matching* problem [141]. The majority of the map matching algorithms focused on the situation when the sampling rate is high (e.g., one sample every 10 seconds) [149]. To solve the map matching problem for low sampling rate vehicle trajectories, Yuan et al. [149] proposed an algorithm that uses sequential samples to estimate the speed and direction of travel, which improved the matching accuracy. The real-valued GPS coordinates were thus converted to discrete road segments to facilitate further processing.

Despite the success of the T-drive project, smart navigation and travel time prediction remain challenging due to various factors such as weather, special events, road works and accidents [140]. T-drive focused on identifying frequent routes traversed by taxi drivers. However, it did not con-

sider the impact of special road events on the selection of travel routes. Nor did it explicitly compare the difference in travel routes before and after an event. In Chapter 4, we propose a trajectory mining algorithm that infers the significant changes in travel routes under the impact of an event. Such an algorithm can potentially assist the search for alternative routes should the designated travel route fail to achieve the desired travel time, as well as providing information to the authorities about the extent of the impact of the event.

2.4 Trajectory Mining for Healthcare at Home

Using smart sensing technology and the IoT for patient monitoring in healthcare has drawn much research attention in recent years. Research in this field is largely motivated by the increasing healthcare cost as a result of the ageing population and a shortage of healthcare professionals [63]. By monitoring patients in their home environment, the aim is to develop systems that can (1) collect real-time information on one's health conditions; (2) send data to a medical center; (3) carefully analyse data to identify trends, anomalies, events or emergencies; and (4) give feedback to patients or notify healthcare professionals when necessary [110]. This forms a layer of automation on top of traditional healthcare systems that aims to achieve two objectives: (1) reduce unnecessary demand on healthcare professionals, which reduces cost; and (2) improve responsiveness to emergencies such as falls, heart attacks or strokes.

To address the task of home-based health monitoring, there exist three major challenges: (1) sensor selection and placement, (2) data transmission and (3) data analytics. In the current literature, various types of sensors have been used, which includes Radio-frequency Identification (RFID) tags [68], wearable devices [110], as well as temperature and force sensors [40]. In terms of data transmission, either wireless sensor networks where data are passed from node to node [39, 68] or a more direct client-server communication protocol has been used [40]. The sensing hardware and communication method being deployed is dependent on the actual monitoring task, which in turn determines the data mining task.

Since patients are not constantly observed by medical professionals in home-based monitoring, one of the problems of interest is to identify what activities have been performed by the patient, which may allow healthcare professionals to offer suggestions and feedback on one's daily routine.

An easy solution would be to place cameras around the home, which can be perceived by many as intrusion into their privacy. Therefore, it is necessary to use less intrusive monitoring methods to detect the activities of the patients, and we have identified two types of sensors that can achieve this task, RFID tags and wearable sensors. RFID tags are miniature chips (as described in Section 2.1) that can be embedded into home appliances and furniture to identify their usage states, and a sequence of RFID sensor signals can indicate the movement trajectory of an individual in a house [115]. They can accurately indicate what objects are being interacted with at what time, and passively reflect the user activity. Wearable sensors, on the other hand, capture data directly from the wearer. Common sensors in a wearable unit may consist of accelerometers [89], gyroscopes [41], magnetometers [5], heart rate and blood pressure sensors [39].

RFID sensors are ideal in detecting the interaction between people and the environment. Activities can be detected by observing signals from related objects. For example, signals from the kitchen door, range hood and kitchen bin may indicate the activity "cooking". However, until the wide-scale adoption of smart appliances where each appliance can communicate via a wireless network, RFID tags must be attached to each object individually, which can be troublesome. Moreover, if the person is performing activities that do not involve external objects, RFID tags will fail to identify the activity.

Wearable sensors, on the other hand, can be used to detect activities whenever they are worn. As more sensors are being embedded into smart watches and smart phones, it is possible to gather data from these sources, which can reduce the potential impact on the set-up cost and quality of life of the user. In this case, the data captured are motion trajectories that reflect the movement of a part of the body. Classifying such data into a number of activities can be more difficult than using RFID data as human bodies move continuously during active periods. In Chapter 5, we address the problem of action recognition with continuous wearable sensor trajectories by proposing a feature construction method and a classification algorithm.

2.5 Summary

In this section, we have introduced the background in relation to trajectory mining in the context of the Internet of Things. We have addressed trajectory mining in community, transport

and healthcare applications. The detailed literature review for existing data mining techniques in each application will be included in each of the following three chapters.

Chapter 3

Recommending Trips by Combining User Trajectories with Smart Sensing

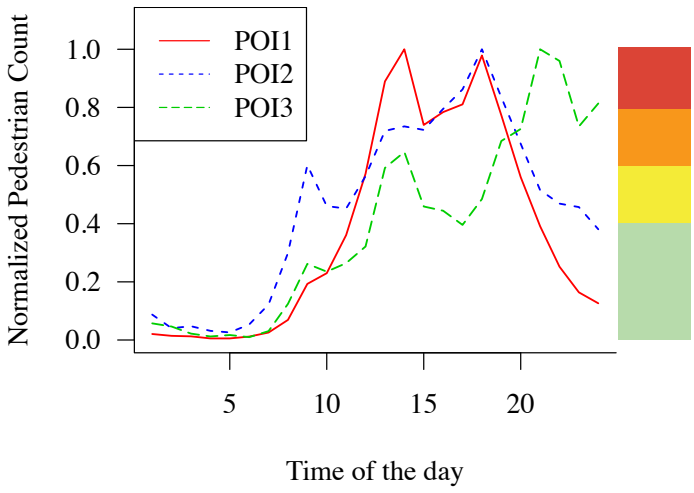
There has been a growing interest in recommending trips for tourists using location-based social networks. The check-ins and geo-tagged photos submitted by a user on a social networking site can be regarded as trajectories that reflect the popularity of the points-of-interest (POI) being visited and the personal preference of the user. The challenge of trip recommendation not only lies in searching for relevant POIs to form a personalized trip, but also selecting the best time of day to visit the POIs. Popular POIs can be too crowded during peak times, resulting in long queues and delays. In this chapter, we propose the Personalized Crowd-aware Trip Recommendation (PersCT) algorithm to recommend personalized trips that also avoid the most crowded times of the POIs using trajectories and pedestrian sensing data. We model the problem as an extension of the Orienteering Problem with multiple objectives, which includes user interest, POI popularity and crowdedness. We solve the objectives using a method based on the Ant Colony Optimisation algorithm. We evaluate our PersCT algorithm using foot traffic information obtained from a real-life pedestrian sensor dataset and user travel histories extracted from a Flickr photo dataset. We show that the trips recommended by our algorithm out-perform several benchmarks in achieving a balance between conflicting objectives such as satisfying user interests while reducing the crowdedness of the trips. The publication arising from the work in this chapter is paper P1.

3.1 Introduction

LOCATION-BASED social networks (LBSNs) have become a rapidly developing field in the last few years. Users of LBSN services may frequently publish their locations online, which generates a large amount of trajectory data. The use of these trajectories has been investigated in several applications, such as friend recommendation [34] and restaurant recommendation [82], to

(a) Trip 1

(b) Trip 2



(c) Pedestrian Volume

Figure 3.1: An example of two trips planned manually in the City of Melbourne with three POIs. (a) Trip 1: maximizing interest. (b) Trip 2: avoiding crowds. (c) Normalized pedestrian volume from [2]. The colors of the legend correspond to the colors of the pins in (a) and (b), which reflect the crowdedness of the POIs.

improve the quality and relevance of query results. One application heavily influenced by LBSNs is trip recommendation for tourists. Mobile-based pocket tour guides have been deployed for small scale applications like museum tour guides [21] or large city guides [136]. Despite these successes, trip recommendation is still a non-trivial problem due to the following challenges:

1. Relevant Points-of-Interest (POIs) must be selected from a large collection of POIs. A naive approach is to select the top k most relevant POIs and list the results, as in POI recommendation using Collaborative Filtering (CF) [145][152]. However, organizing a trip from such a list can yield a solution that is far from optimal, as the POIs can be spatially distant and the user might not have enough time to visit all POIs in a single trip.

2. Constructing an optimal solution requires all permutations of the POIs to be computed, which is an NP-hard problem. The computational cost will be prohibitive even for a small number of POIs.

3. For POIs of different categories, different peak hours may apply. Visiting a POI during the peak time may result in a long wait time, poor service and sometimes a higher price. Previous studies on temporal POI recommendation have focused on recommending popular times at a POI [151, 152] rather than avoiding crowded times.

Various trip recommenders have been proposed to recommend personalised trips by exploiting user-submitted trajectories [87][156]. However, previous studies have failed to consider that some POIs may satisfy a user's interest but can be too crowded at times. Fortunately, pedestrian traffic data from sensor deployments [2] is making it possible to refine trip recommendations based on how crowded places are at different times of the day. In this chapter, we propose the personalised crowd-aware trip recommendation (PersCT) framework, which recommends personalized trips that avoid crowded areas to users. We assume that users have no knowledge of the city they will be visiting, and we aim to achieve two objectives: (1) selecting a few POIs from all the POIs in a city and (2) organising the POIs into a trip for a user. Our approach can also handle the case when a user already knows which POIs to visit, in which case only simple modifications are needed to recommend the best route with low crowdedness. To illustrate an example, we show two trips manually planned with three POIs in the City of Melbourne, Australia (Figure 3.1).

Figure 3.1(a) and 3.1(b) show the trips marked on a map and Figure 3.1(c) shows the normalized pedestrian foot traffic captured by nearby sensors at the POIs (details in Section 3.5). The colors of the POIs in Figure 3.1(a) and 3.1(b) correspond to different crowdedness levels in Figure 3.1(c). POI1 (David Jones) is a shopping mall and the peak foot traffic is between 12:00 and 17:00. POI2 (Flinders Street) is near a famous laneway, a visitor center and a train station/historical site whose pedestrian volume peaks at 17:00. POI3 (Lygon Street) is a popular Italian dining destination and the peak time occurs after 20:00. Assuming a user leaves the airport coach terminal at location 1 at 12:00 and the destination is a hotel at location 5, trip 1 might be a typical trip where the user interest is maximized: the user visits POI2 between 13:00 - 16:00 for sightseeing, POI1 between 16:00 - 19:00 for shopping and POI3 between 19:00 - 21:00 for dining and relaxation. However, the crowdedness of this trip would be very high as shown in Figure 3.1(c). In contrast, trip 2 offers a different plan: visit POI3 between 13:00 - 16:00 for lunch and sightseeing, POI1 between 16:00 - 19:00 for shopping and POI1 between 19:00 - 21:00 for dinner and a walk by the Yarra river. By slightly shuffling the order of the visits, crowds can be avoided while the user interest can still be satisfied. (Lunch at popular restaurants is also usually cheaper than dinner!)

With only three POIs, it is possible albeit difficult to manually plan a trip that avoids crowds. However, the number of POIs in a city is typically much larger than three, and the search space increases exponentially. Therefore, algorithmic approaches must be efficient to perform trip recommendation. In this work, we define the problem as an extension of the Orienteering Problem (OP) [56], which models each POI with a profit score and finds the trip that maximises the total profit while satisfying certain time constraints. We define the profit to be a time-varying function that combines POI popularity, user interest and crowdedness of the POIs at various times of the day. With this information, we formulate the personalized crowd-aware trip recommendation problem as a multi-objective time-dependent Orienteering Problem. We propose the PersCT algorithm, which is an extension of the Ant Colony Optimisation (ACO) metaheuristic, to solve the problem. Using this algorithm, trips that avoid the crowds while still satisfying user interests can be found efficiently. Our main contributions are:

- We formulate the personalized crowd-aware trip recommendation problem.
- We combine multiple objectives and propose the PersCT algorithm that can efficiently find a solution to balance conflicting objectives such as user interest and crowdedness of trips.

- We evaluated the effectiveness of our algorithm using a Flickr photo dataset and pedestrian count dataset from pedestrian sensors, both in the City of Melbourne.

The rest of the chapter is organized as follows. Section 3.2 discusses related work in trip recommendation and planning. Section 3.3 presents relevant definitions and formally defines the problem. In Section 3.4 our trip recommendation framework is discussed. In Section 3.5 we present the evaluation and discussion.

3.2 Related Work

In this section, we review existing work related to the trip recommendation problem. Suggesting interesting locations to tourists is a widely-studied problem in the data mining and operations research community. The work in this field can be classified into algorithms that (1) recommend individual POIs and (2) recommend trips that optimise one or more objectives. The former class of algorithms focus more on personalisation whereas the latter class mainly tackle the route optimisation problem under different constraints. In this work, we address both personalisation and optimisation, and therefore, we split previous studies into two sub-sections: (1) POI recommendation and (2) itinerary recommendation.

3.2.1 POI Recommendation

The problem of recommending individual POIs is similar to the general class of recommender systems problems, which has been a field of intense research in the past decade [145]. Several applications have seen the early success of recommender systems such as recommending books and CDs [88], movies [96] and news [20]. As smart phones have become widely-available in recent years, POI recommendation has emerged as an important problem in LBSN [145]. POI recommendation algorithms learn the preferences of a user and suggest attractions that have not been visited. The presentation of the results is usually a list and items in the list do not necessary relate to one another. This is different from our objective where the results must be presented as a trip. However, a part of our objective is that the trips should be personalised, and thus we review POI recommendation to compare and find the most suitable algorithm to use, rather than proposing a new POI recommendation algorithm.

Collaborative filtering (CF) is one of the most well-known and widely-used techniques in POI recommendation [134] with three common approaches: user-based CF, item-based CF and matrix factorisation [145]. For a POI recommendation problem, the input data consist a list of N users $U = \{u_1, u_2, ..., u_N\}$ and M POI locations $I = \{P_1, P_2, ..., P_M\}$. For each location, a user u_i has a visit count and may include the time and date of the visits. Counting the number of visits for all locations of a given user will create a user-POI vector, and a user-POI matrix can be obtained when all user-POI vectors are stacked together.

User-based CF (UBCF) works by finding the top k most similar users $u_1, .., u_k$ to a particular user u_i, and then aggregates the rating scores (or visit counts in our case) to produce the recommended locations [144]. The top k similar users are found by computing a similarity measure $sim(u_i, u_j)$, and a number of similarity measures have been proposed in the literature such as Pearson correlation scores, cosine and Jaccard similarity index [145]. POI ratings from the top k users are usually aggregated by a sum weighted by the similarity measure. In addition, geographical influence is often exploited in POI recommendation. In [144], the authors model the distance between two check-ins as a power distribution. This comes from the intuition given by the first law of geography, which states "everything is related to everything else, but near things are more related than distant things" [132]. The next POI to recommend is weighted by the likelihood of a visit generated by the power distribution, and the results are significantly improved over non-weighted UBCF.

Item-based CF (IBCF) works slightly differently as the similarity between each POI (the "items") are computed. The name originated from e-commerce where the problem is to recommend items to users. Assuming that users take the rows in the user-POI matrix and the POIs take the columns, UBCF computes the similarity between each row, whereas IBCF computes column similarity scores. Item-based CF has been used in POI recommendation, although it has been reported that UBCF outperforms IBCF [144]. This may be due to the fact that the dataset being evaluated in the experiments contains more POIs than users, and thus more values are missing if the data are viewed from the columns than from the rows. For large scale deployment where there are millions of users and only thousands of POIs, IBCF would perform better [145].

Matrix factorization (MF) has become a commonly used technique in recommender systems since the success of the Netflix competition [76]. Berjani et al. [18] proposed a regularized matrix

factorization model to recommend POIs to users. The visit history of each user is converted to a sparse vector that counts the number of times each POI is visited, and vectors from all users are stacked to form a N-by-M user-POI matrix C, where N is the number of users and M is the number of POIs. For each user, the number of unvisited POIs usually exceed the visited ones, and the problem is thus converted to a missing value imputation problem. As the raw POI visit counts can be unbounded, they must be converted to either binary or binned counts. Binary conversion means to convert the matrix into a binary matrix where zeros are unvisited and ones are visited POIs. Binned counts can be produced by setting a maximum count threshold for the maximum amount of user interest to be defined, and then use equally spaced bins to quantize the matrix. Then the user-POI matrix is decomposed into two low rank matrices with latent (hidden) features, the K-by-N user-feature matrix U and the K-by-M POI-feature matrix V, where K is the number of latent features. The latent features represent hidden user preferences that are not observable in the user-POI matrix, such as categories of POIs, cost, etc. The value K is usually chosen via experimentation and in most cases it is less than 30. As in the case of Singular Value Decomposition [57], the first latent feature captures the dimension with the largest variance in the data, which represents the most important reason why people choose a POI. The factorization is usually performed by the stochastic gradient descent procedure [76]. First, values in the low rank matrices are randomly initialised. The user-POI matrix is multiplied back using the low rank matrices and the errors between the estimated POI scores and observed POI scores in the training data are obtained. To reduce the error in the estimation, the gradient of the error function is computed, which is then added to the current low rank matrices. The user-POI matrix is multiplied back again and the procedure repeats until the error converges [18]. The advantage of MF-based methods is the relative improvement in performance when dealing with sparse data. However, as MF is a global method, every change in the input data requires re-computation of the entire factorization process, which can be computationally intensive in a POI recommendation system that requires frequent updates as users may check-in several times a day.

By reviewing the above work, we can conclude that user-based collaborative filtering is the most suitable algorithm to be used in our trip recommendation system due to its speed, simplicity and flexibility. In addition to the above papers, we found several recent studies that utilise collaborative filtering to recommend time of visit as well as locations [151, 152]. The approaches used

can find popular times to visit each POI whereas in our work we discover times when the POIs are not crowded. Moreover, in previous studies, there is no overall time budget constraint, and travelling time is not considered. In our work, we find relevant POIs as well as ordering the POIs into a trip to satisfy constraints such as the maximum allowed trip time.

Collaborative filtering suffers from the "cold-start" problem, which occurs when a completely new user tries to use the system. As the user does not have any previous visit histories, the recommender will not be able to find similar users to recommend POIs. This is an intrinsic deficiency of CF-based algorithms. In our work, we combine the popularity of POIs and the estimated user interest into a single score so that when the user visit history is unavailable, the popularity score is used to recommend and schedule the trips. Consequently, new users will be able to use the system even if they have no past visit histories.

3.2.2 Itinerary Recommendation

Itinerary recommendation is the problem of recommending the complete trip, or itinerary, to the user. This problem is usually formulated as the Orienteering Problem (OP), whose name is derived from the sporting game of orienteering [133]. It is formulated as follows: given the origin, the destination and a utility function for each node, find a path on a subset of the nodes of a graph with the maximum utility that satisfies certain constraints. This problem has been extensively studied by the data mining and operations research community with different foci. The former focus was more on identifying insightful features in datasets to improve trip personalisation and user modelling, while the latter was focussed more on heuristic algorithms that can solve the optimisation problem with a better solution.

One of the early state-of-the-art itinerary recommenders was proposed in [37]. The authors formulate the problem as a classic orienteering problem. The objective is to recommend as many relevant POIs in a trip as possible while still satisfying the time limit given by the user. There are two core procedures, calculating the time of visit and trip scheduling. As users are often unaware of how long they should spend at each POI or the transit time, it is necessary to estimate these values from available data. The authors use geo-tagged Flickr photos and extracted trips between the POIs. The difference between the time at two consecutive POIs is taken, which is used as the POI visit time plus the transit time. The trip planning is completed using a recursive

greedy algorithm [30]. The algorithm splits unvisited nodes into an upper and lower half. The next destination is assumed to be in the first half and the algorithm determines the amount of the budget constraint consumed by visiting that location. Then the recursion is called on each half until the budget is exhausted. This approach is also capable of recommending multi-day itineraries. However, the authors have not addressed the personalisation issue, and all users would obtain the same result when positioned at the same location. In addition, the time dependency of the recommendation is not considered and users will obtain the same result in the morning and afternoon, which is not ideal. In our work, we personalise the trip as well as consider the time dependency in the trip recommendation, which can be more realistic and accurate.

In [92], the authors estimate the attraction score of a POI using a user-based CF. The attraction score is a fusion of CF ratings given by strangers that have similar visit profiles as well as the friends of the current user. For each POI, the popularity at various times of the day is also factored into the attraction score, and the objective is to find trips that have the maximum attraction score for a certain user. Consequently, the objective is a time dependent function. The authors propose an exhaustive pruning and bounds check algorithm based on the apriori algorithm in itemset mining [26] to find optimal trips under multiple constraints. The intuition is that if a sub-sequence has violated the constraints, its super-sequences are also invalid and can be pruned out. Despite the effectiveness of the approach, the computational cost remains relatively high. For a trip with 8 hours, the computation time can be up to 60 seconds without parallel computation. This is prohibitive in real-time user applications. A similarly effective but computationally expensive algorithm was proposed in [87], where the authors used integer programming to optimise trip recommendation. The amount of time that a user should spend at each POI is computed by evaluating the time spent on venues with the same category. These values are then converted to a time-based user interest score, where the raw time is normalized by the maximum value into the range [0,1]. The objective function is constructed by a weighted sum of the time-based user interest score and POI popularity scores derived from the whole dataset. The disadvantage of this approach is that two users with the same preference for each category will obtain the same results at the same location, whereas an ideal algorithm will personalise the trips for each individual. Moreover, the computaional complexity prevents the algorithm from giving a solution for more than 50 POIs, and for large cities with many POIs, this algorithm cannot be used. Our work differs from the above since we propose

a meta-heuristic algorithm that has quadratic time complexity and can be run very efficiently. As can be seen in Section 3.5, the running time of our algorithm is less than one second, which is sufficient for real-life trip recommendation.

Tourist trip recommendation is also a well-studied problem in the operations research community, where the focus is on the theoretical aspects of solving the Orienteering Problem (OP) with various constraints. The OP has been proven to be NP-hard [48] and a survey of the trip planning algorithm with an OP formulation can be found in [53]. The theoretical aspects include whether the graph is undirected [13] or directed [101] and whether a trip includes the terminal node [32]. In our work, we assume the graph is undirected as our mode of transport is assumed to be walking. We further assume that the starting and terminal nodes are given. In addition, we have a time-dependent objective function, which is different to most of the papers with an OP formulation. This is an important property as the algorithms that take a classic integer programming formulation cannot be directly used and is difficult to take into account the time-dependency. Therefore, we only review papers that have solved a time-dependent variation of OP.

In [137], the authors propose a model to estimate the time-dependent travel speed between two POIs. The travel speed is computed based on the time of the day and the location of the road. In the busy city center during the morning and afternoon peak, the travel speed is usually low whereas the travel speed tends to be high in a rural area. The detailed speed model can be found in [69]. An Ant Colony Optimisation (ACO) algorithm is used to solve the problem [48]. Although the authors use the same ACO framework as our work, the objective of their approach is to minimize the travel time. In contrast, we have multiple objectives such as user-interest as well as the distance covered, which presents a more difficult challenge. In addition, the authors assume that early starters will also arrive early at their destinations, which may not be true in real-life scenarios due to traffic lights. In [136], a city trip designer was proposed to plan tours that consider the opening and closing times of a POI. This problem is called the Orienteering Problem with Time Window (OPTW), a popular variant of the OP. The authors ask the user to specify their level of interest in the types of venues and provide a score between 0 to 5. The user can also input some keywords to specify their interest and an estimate of the overall interest score is computed using a combination of venue types and keywords. The solution of the problem is found using a Greedy Randomized Adaptive Search, which is an iterative algorithm. At the start

of every iteration, a new greediness value between 0 and 1 is generated randomly from a uniform distribution. This value determines the strategy of search where a value of 0 is completely greedy (like the nearest neighbour algorithm) and a value of 1 is completely random. Next, using the start and end of each trip, a candidate list of possible visits is generated before being filtered using a heuristic function. The best trip is stored by the algorithm as the final result. This approach is simple and efficient algorithmically. However, it can be cumbersome for each individual user to input the list of keywords and preferences for trips to be recommended. In [61], the authors propose a mixed-integer programming algorithm to recommend trips for theme park visitors. The time dependency caused by uncertain wait times at the ticketing office has also been addressed. In [54], a time-varying travel cost was modelled and trips were recommended for a group of users. These two papers mainly focused on the trade-off between efficiency and optimality of the solution rather than the relevance of the trips to the users. Moreover, the trips are not personalized and the same origin/destination pair will result in the same trip for all users.

In addition to the above work, the authors of [23] formulated the problem as a maximum coverage problem and solved the trip recommendation using POI popularity and user interest. Chen et al. proposed a personalized trip planner using digital footprints [31]. More recently, in [70], personalized travel sequences in different seasons were recommended by merging textual data and view point information extracted from images. The most recent work relevant to ours is [156], where the authors proposed a tree-based algorithm to solve the personalized trip recommendation problem. Our work differs from the above as we additionally focus on balancing the crowdedness of a trip with other objectives via the integration of two types of data, pedestrian sensing data and user submitted trajectories. To the best of our knowledge, our work is the first to combine crowdedness, user interest and POI popularity in recommending trips.

3.3 Preliminaries

In this section, we give the necessary definitions and formally define the problem.

Definition 1: POI Graph. For a region with W POIs, we construct an undirected complete weighted graph G with W nodes being the POIs, and edge weights $e(i, j)$ being the travel time between two POIs. For simplicity, we assume travel time is symmetric, the mode of transport is

walking and the distance to travel between two nodes is a straight line, as in [87][1]. Travel time between two POIs can then be computed by dividing their distance by the speed of walking. For a certain user, a trip $R =< O, P_1, P_2, ..., P_M, D >$ is a non-cyclic path on the graph G where $O =$ origin, $D =$ destination and P_i, $i = 1,...,M, M \leq W-2$, are the POIs visited. The time that the user arrives at each POI is $T =< T_O, T_1, ..., T_M, T_D >$. The duration that the user spent visiting each POI is $Du =< Du_O, Du_1, ..., Du_M, Du_D >$. The travel history H of a user is $H =< R_1, R_2, ..., R_L >$, where $L \geq 1$.

Definition 2: Time Cost. The time cost $C(R)$ of a trip R is the sum of all travel times and the time spent visiting each of the POIs of a trip:

$$C(R) = \sum_{i=1}^{M-1} \sum_{j=2}^{M} e(R(i),R(j)) + \sum_{k=1}^{M} Du_k,\ j = i+1 \tag{3.1}$$

where $R(i)$ is the i^{th} POI in trip R.

Definition 3: POI Popularity. For POI i, we calculate the POI popularity $Pop(i)$ by counting its occurrence $Ocr(i)$ in the travel history of all users and normalize the values to [0,1]:

$$Pop(i) = \frac{Ocr(i)}{max(Ocr(j, j \in W))} \tag{3.2}$$

For example, if there are 3 POIs in total, and $Ocr(1) = 10$, $Ocr(2) = 50$, $Ocr(3) = 20$, then $Pop(1) = 0.2$, $Pop(2) = 1$, $Pop(3) = 0.4$.

Definition 4: User Interest. We define the interest score $Int(u,i)$ of a user u to a POI i as the similarity between POI i and the past visiting history of u. More on this is given in Section 3.4.2.

Definition 5: POI Crowdedness. For POI i, we define the crowdedness at time t as the foot traffic volume U divided by the maximum foot traffic detected during a certain period of time T_p:

$$Crd(i,t) = \frac{U(i,t)}{max(U(i,t \in T_p))} \tag{3.3}$$

For example, a sensor reported the pedestrian foot traffic at a POI to be $< 100, 200, 300 >$ at 9:00, 10:00 and 11:00. The maximum pedestrian volume observed for this POI is 500. Therefore the crowdedness during these three hours is $< 0.2, 0.4, 0.6 >$.

[1] Other travelling distance functions and modes of transport can also be incorporated into the problem

3.3.1 Problem Definition

Given a user u with an origin O, a destination D, a start time t and a time budget c, we recommend a trip R such that the following objective function is maximized:

$$max\left(\sum_{i=1}^{W-1}\sum_{j=2}^{W} x(i,j,t)pr(u,j,t)\right) \tag{3.4}$$

where the profit, $pr(u,j,t)$, is defined as

$$pr(u,j,t) = \frac{(Pop(j)+Int(u,j))^{\gamma}}{Crd(j,t)}, \gamma > 0 \tag{3.5}$$

The function $x(i,j,t)$ equals to 1 if P_i and P_j are consecutively visited POIs in R, and $x(i,j,t)$ $= 0$ otherwise. The parameter t is the arrival time at POI j. We optimize the objective function such that the following constraints are satisfied: (1) the total time cost is no greater than the time budget; (2) no POIs are visited more than once; and (3) the trip always starts at the origin and finishes at the destination.

Equation 3.4 is a multi-objective time-varying function that merges POI popularity Pop, user interest Int and POI crowdedness $Crd(i,t)$ information at the time of the visit. We aim to maximize Pop and Int while minimising Crd. A positive parameter γ is used to control the bias towards popularity and interest or crowdedness. A large γ will place more weight on selecting relevant POIs while a small γ will encourage searching for less crowded locations. We show in Section 3.5 the effect of tuning γ on the performance of the proposed algorithm. Based on the above definitions, we present our trip recommendation framework in the following section.

3.4 Trip Recommendation

In this section, we present our trip recommendation framework.

3.4.1 Overview

There are three main stages to our trip recommendation framework, namely POI popularity modelling, user interest modelling and the PersCT algorithm. Figure 3.2 shows an overview of

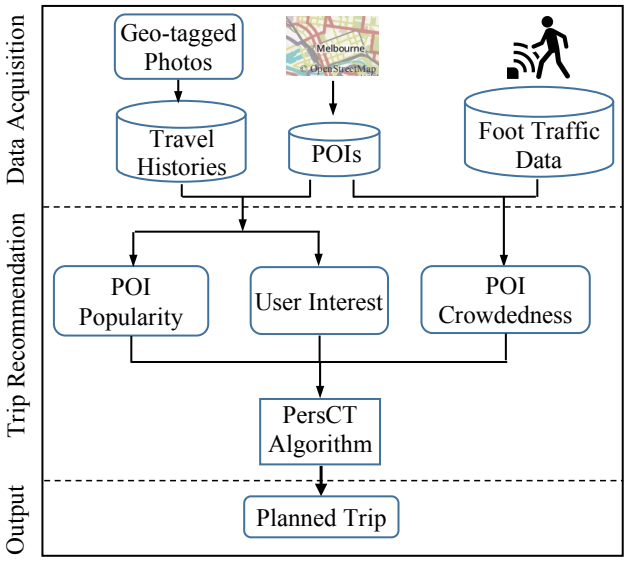


Figure 3.2: Overview of our approach

our system. POI popularity is given by Equation 3.2, which defines a normalized popularity score for each POI by counting its occurrence in the travel histories of all past users. The reason for including such information is that when a new user with no travel history is using the system, popular locations can still be used to find a solution. This addresses the "cold start" problem frequently faced by user modelling algorithms such as collaborative filtering. The output of Stage 1 is a vector of POI popularities, Pop. The second step is the modelling of user interest. For users who have visited at least three POIs in the past, we infer the user interest for the unvisited POIs by performing User-Based Collaborative Filtering (UBCF, more on this in the next section). The result of UBCF is a vector of scores Int corresponding to user interest for each POI. The sum of Pop and Int are used as part of the objective function that is to be maximized (Equation 3.5). The third stage is to combine foot traffic information with Pop and Int to obtain the objective function for the multi-objective time-dependent orienteering problem, which is then solved under various constraints as defined in Section 3.3.1. Since OP is NP-hard, our trip recommendation problem is also NP-hard. In addition, the crowdedness objective is a time-varying function, which is difficult to optimize using exact algorithms like dynamic programming or off-the-shelf optimisation solvers. To this end, we adapt and extend the Ant Colony meta-heuristic to find a solution (Section 3.4.3 to 3.4.5).

3.4.2 Modelling User Interest

We implement a user-based collaborative filtering (UBCF) algorithm to personalize user interest based on previous travel histories due to its simplicity and proven performance in POI recommendation [151]. We create a user-POI count matrix H where the element H_{ij} represents the number of times the user u_i has visited the POI P_j. As counts can take the range $[0,\infty]$, H must be normalized to have a finite maximum. To this end, we set a maximum value for the number of visits and all visit counts larger than 5 are limited to 5. This is because we consider all frequently visited POIs as equally interesting to a user, which simplifies the problem. We now call the H matrix a rating matrix as we model the entry H_{ij} as a rating given by user u_i to POI P_j. As a user typically only visits a small subset of all POIs, the H matrix contains mostly missing values and the next step is missing value imputation. A core assumption in UBCF is that similar users will visit similar POIs. Therefore, the rating user u_i gives to POI P_j is calculated as an aggregation of the ratings given by the k most similar users of u_i:

$$r_{u_i,P_j} = aggr(r_{u'_i,P_j}),\ u'_i \in U$$

where U is the set of the k most similar users. The cosine similarity was used as the similarity measure between user u_i and u'_i:

$$sim(u,u') = \frac{\sum r_{u,i}\ r_{u',i}}{\sqrt{\sum r_{u,i}^2}\sqrt{\sum r_{u',i}}^2} \tag{3.6}$$

In addition, we selected the mean function as the aggregation function to average the ratings given by the k similar users, and we set the number $k = 15$. The ratings estimated by UBCF are used as user-specific interest scores $Int(u_i,P_j)$ for every unvisited POI of each user. Since this only needs to be computed once, the running time is negligible.

3.4.3 The Ant Colony Meta-heuristic

In this work, we extend the Ant Colony Optimization Meta-heuristic (ACO) [48] to solve the multi-objective time-dependent optimization problem. For completeness, we briefly introduce the ACO algorithm in this section (Algorithm 3.1). ACO is an iterative heuristic where software

agents, or the ants, search for solutions to a given optimization problem. The ants are involved in two procedures: (1) heuristic solution construction, and (2) iterative pheromone trail update. In the solution construction phase, a bottom-up approach is used to find the next destination of an ant. A probability score is computed for each unvisited location using a specifically designed heuristic function for each particular problem. The search function can be as simple as a nearest neighbour search. As the scores are probabilities, the sum of the scores equal to one and the higher the score, the more likely it is to select a location as the next destination. The scores are copied into an array to form "bins" and a random number in the interval [0,1] is generated. The "bin" that the random number falls into is selected as the next destination. In the pheromone trail update phase, the ants "communicate" with one another by updating a trail matrix TM. The basic idea is that locations frequently visited by ants should have a higher likelihood of being a part of a good solution, and the trail matrix stores the frequency of visits and passes this on to the next iteration. The trail update involves two parts, online and offline update. In online update, after finding the next destination P_{i+1}, an ant immediately updates $TM(P_i, P_{i+1})$, reinforcing locations with large visiting probabilities. In offline update, only the best ant updates the trail matrix with every path visited in its trip. After the last iteration, the trip of the best ant is output as the result. In addition to the two steps mentioned above, an optional local heuristic search step can also be used with ACO. An example is swapping the adjacent two nodes or randomly inserting a location into the solution. These optional local heuristics can find a better local optimum if designed properly.

3.4.4 PersCT Algorithm

In this section, we discuss the details of our PersCT algorithm in three subsections: heuristic search function, local search and trail update strategy.

Heuristic Search Function

For each ant, the next POI to visit is found using a heuristic probabilistic search function. The original Ant Colony algorithm suggested a search function in the following form:

$$Prob(i,j) = \frac{tr(i,j)^a pr(j)^b}{\sum_i^W \sum_j^W tr(i,j)^a pr(j)^b} \tag{3.7}$$

Algorithm 3.1: Ant Colony Optimization Metaheuristic

```
Data: G: a graph
Result: P: a path on G
begin
    Initialize pheromone trail over G
    while termination condition not met do
        Initialize ants
        //construct solution
        while ∃ ant ∈ ants not finished do
            for ant ∈ ants do
                Find next node
                // online trail update
                Update pheromone trail
                Check if ant has finished
        Update global best ant
        Perform local search
        // offline trail update
        Update pheromone trail with global best ant
    return path P traversed by the best ant
```

where j belongs to the set of all unvisited POIs, $tr(i,j)$ is the trail matrix and $pr(j)$ is the profit function. This follows the intuition that trails frequently visited by ants are likely to be the ones that have higher utility scores. The profit score can be as simple as the inverse of the distance to the next node or a more complex function. The denominator normalizes the numerator into a proper probability score where the sum is taken over the set of all unvisited nodes. The parameters a and b set the preference for using trails versus exploring new nodes [48]. In this work, we incorporate the following into the PersCT framework:

- popularity and user interest objectives
- time-varying crowdedness information
- a distance re-weighting function.

Our search probability function is defined as the following:

$$Prob(i,j,t) = \frac{tr(i,j,t)^a pr(u,j,t)^b f_{dist}(i,j)}{\sum_i^W \sum_j^W tr(i,j,t)^a pr(u,j,t)^b f_{dist}(i,j)} \tag{3.8}$$

The search function is a product of three terms: (1) trail left by past ants, (2) computed profit score of unvisited POIs for this user, and (3) distance re-weighting. The pheromone trail matrix $tr(i,j,t)$ stores the contributions from ants in the previous iterations. Since we must consider the time-dependence of profit scores, the trail contains a time dimension rather than a purely spatial formulation.

The profit function $pr(u,i,t)$ takes into account (1) POI popularity as defined in Equation 3.2, (2) user interest as defined in Section 3.4.2, and (3) time varying crowdedness. We define a distance re-weighting function f_{dist} that re-weights the visiting probability based on the distance to the current POI and the final destination. Let i be the current POI, j the next POI to select from unvisited POIs, and D the final destination, then the probability of visiting j is re-weighted by a factor f_{dist}:

$$f_{dist}(i,j) = \frac{1}{\exp(dist(i,j) + \frac{dist(j,D)}{dist(i,D)})} \tag{3.9}$$

The search function is re-weighted such that these POIs are preferred: (1) POIs that are close to the last visited POI, (2) POIs that are closer to the final destination than the last visited POI. Rule 1 is based on the first law of geography, which states "everything is related to everything else, but near things are more related than distant things" [132]. Rule 2 is based on the observation that people tend to move towards the final destination and visit POIs on the way rather than move away from it. Figure 3.3 shows a histogram of the ratio $\frac{dist(i+1,D)}{dist(i,D)}$ for a Flickr dataset in Melbourne, Australia (see Section 3.5). Approximately 68% of the ratios are less than 1, and the vast majority are less than 1.5, which supports the re-weighting scheme. In Section 3.5, we show that distance re-weighting is effective in recommending trips.

Local Heuristic Search

We apply a local heuristic search after the conclusion of each iteration when all ants have reached their destination. The best ant is updated and we generate a temporary ant to store the best ant. We randomly swap all pairs of the visited POIs of the best ant. If a better solution is found, then we update the temporary ant with current best ant. Otherwise we undo the previous swap and re-start from the previous POI. This is to ensure that there exist no overlapping edges as the solution by ACO provides no such guarantee. Next we randomly swap an unvisited POI with

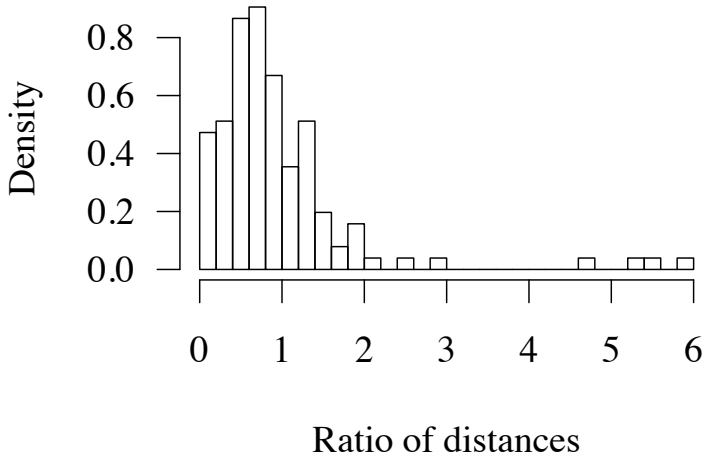


Figure 3.3: Histogram of the ratio *dist(i+1,D)/dist(i,D)*.

one POI in the trip. This scheme can increase the search space with little computational cost and improve the quality of the solution.

Trail Update Strategy

During each iteration, we implement the online trail update rule as the following:

$$trail(i,j,t) = (1-\rho)trail(i,j,t) + \frac{\rho}{W} \tag{3.10}$$

where W is the number of unvisited nodes in the POI graph. For both online and offline trail update steps, previous trails are first "evaporated", or multiplied by a constant $(1-\rho)$ where $\rho \in [0,1]$ [48]. This is because when "bad starts" occur in the first few iterations due to the random nature of the algorithm, the solution space will be confined to a local optimum without trail evaporation. Limiting the influence of previous ants can improve the overall quality of the solution. The amount $\frac{1}{W}$ is deposited on the trails of new ants as the influence of the current trail should be inversely proportional to the number of unvisited nodes.

After all ants have stopped, the global best ant is updated by finding the ant with the maximum objective score (Equation 3.4). The pheromone trails that the best has visited are updated using

Algorithm 3.2: PersCT

Data: O = origin, D = destination; T = time limit; $Cost$ = travel time matrix; Pop = popularity array; Int = user interest array; Crd = crowdedness array; P = list of POIs
Result: R = Best trip

```
1  begin
2      trail ⟵ matrix of 1.0, bestAnt ⟵ ∅
3      while iteration < maxIterations do
4          Initialize ants with < O,D >, stopped ⟵ ∅
5          while stopped ≠ ants do
6              for ant ∈ ants do
7                  if ant ∉ stopped then
8                      from ⟵ last visited POI
9                      depTime ⟵ departure time at from
10                     SelectNextPoi(ant);
11         Update bestAnt
12         Update trail with bestAnt (Equation 3.11)
13     while iteration < maxIterations do
14         Swap bestAnt.i with bestAnt.j
15         Update bestAnt.obj (Equation 3.4)
16         if bestAnt.obj < tempAnt.obj then
17             Swap bestAnt.j with bestAnt.i
18     return bestAnt.trip
```

the offline update rule as in [48]:

$$trail(i,j,t) = (1-\rho)trail(i,j,t) + \rho pr(bestAnt) \tag{3.11}$$

where $pr(bestAnt)$ is the profit score of the trip produced by the best ant (Equation 3.4). The same evaporation procedure is used as in the online trail update by multiplying a constant $(1-\rho)$ where $\rho \in [0,1]$. The profit score of the best ant is deposited to each trail edge that the ant has visited, increasing the likelihood of ants visiting the same edge in the next iteration.

3.4.5 Algorithmic Implementation

In this section, we describe the implementation details of the PersCT algorithm, which consists of the following three steps.

Algorithm 3.3: SelectNextPoi

```
Data: ant; P = list of POIs; T = time limit
Result: ant
1  begin
2      prob ⟵ ∅
3      for p ∈ P do
4          if p ∈ unvisited then
5              prob[p] ⟵ Equation 3.8
6      v ⟵ random ∈ [0,1], s ⟵ 0
7      for pr ∈ prob do
8          s ⟵ s + pr
9          if s ¿ v then
10             to ⟵ index of pr in prob
11     if to ≠ ant.Destination then
12         Add to to ant.trip
13         c ⟵ cost of ant from Equation 3.1
14         if c < T then
15             Update ant.obj (Equation 3.4)
16             Update trail with Equation 3.10
17         else
18             ant.stopped ⟵ TRUE
19     else
20         ant.stopped ⟵ TRUE
21     return ant
```

Step 1: Mapping POI visits. We extract user travel histories by mapping geo-tagged photos to the list of POIs. In particular, we map a photo to a POI if their coordinates differ by $< 200m$ based on the Haversine formula [122], which is used for calculating spherical (earth) distances. The sparsity of the user-visit data prevented the use of a smaller mapping radius as many photos would otherwise be classified as not belonging to any POIs. However, for real-life applications in tourism, a granularity of 200m is typically sufficient. We calculate the popularity of POIs and the rating scores of a user to each of their visited POIs using user-based collaborative filtering as described in Section 3.4.2.

Step 2: Calculating crowdedness. Due to practical sensor placement issues, foot traffic sensors may not be placed at the exact POI locations. Therefore, we estimate the foot traffic volume at the location of POIs using a nearest neighbour based estimation method. Initially, we considered using the nearest sensor to the POI as the POI's pedestrian volume. However, some sensors contain anomalies and missing data, which causes difficulties in the later computing stages if ignored. Therefore, we find the three nearest sensors within 200m of a POI, and compute the mean pedestrian volume weighted by the inverse of their distances to the POI. This resolves the missing data problem and also eliminates sensors that are placed too far away from the POIs. As defined in Equation 3.3, we calculate the crowdedness of the POIs at each time instance by dividing the current volume by the recorded maximum volume to normalize the values into the range [0,1]. For each POI, the maximum pedestrian volume is found by scanning through the data in the whole dataset. We then average the crowdedness of each POI using its weekly values at the same time. Although in this work we do not implement a real-time system due to the sensor and visits data being captured on different days, our framework can be readily deployed when real-time data are available.

Step 3: Recommending trips. We compute the travel time matrix as in Equation 3.1. Using POI popularity, user interest, crowdedness, travel time, start time and maximum trip time as input parameters, the tours are generated by giving the current location and the expected destination of the user to the PersCT algorithm (Algorithms 3.2 and 3.3). Lines 3 to 12 in Algorithm 3.2 are the main loop where the solution space is explored. In each iteration, all ants are re-initialized with the given origin and destination. The set $stopped$ contains ants that either have reached their destination or exceeded a time limit (Line 4). If at least one ant has not stopped, then it enters the

inner while loop to search for the next POI to visit (line 8-10), and Algorithm 3.3 is called.

Lines 2-5 of Algorithm 3.3 compute the visiting probability to all unvisited POIs using Equation 3.8. Line 6 initializes a random floating point number v in the range [0,1] and uses it to set a threshold to determine the next POI to. Lines 8-10 compute the cumulative probability score and when the sum is greater than v, to is set to the index of the last value. Lines 11-20 update the objective function of the ant and also perform the trail update using Equation 3.5 and Equation 3.10. If the stopping criterion is met (lines 18 and 20), ant is added to the list of stopped ants and will not search for another POI in this iteration. The function SelectNextPoi then returns the ant.

Line 11 of Algorithm 3.2 updates the global best ant by finding the maximum objective score. On line 12, offline trail update is performed (Equation 3.11). Lines 13-17 perform the swapping local search discussed in Section 3.4.4. Lastly, the trip by the best ant is returned.

3.5 Experimental Evaluation

In this section, we present the experimental evaluation of our framework.

3.5.1 Datasets and Pre-processing

The evaluation was performed using datasets collected in Melbourne, Australia (Table 3.1). A list of 242 POIs in Melbourne was downloaded from [1]. User travel histories were extracted from the Yahoo! Flickr Creative Commons 100M (YFCC100M) dataset [131], which contains 100 million photos and videos taken by real users. Visits within 200 meters of the POIs were kept and the rest were removed. For each user, photos within eight hours were grouped into a trip, producing a total of 3975 tours and 17087 visits.

Flickr Photos		
No. Users	No. Trips	No. Visits
911	3975	17087
Pedestrian Foot Traffic Data		
No. Sensors	Data Rate	Period
41	1/hour	1/May/2015-30/Sep/2015
POI data: 242 POIs		

Table 3.1: A summary of the datasets.

We extracted pedestrian foot traffic from the Melbourne Open Data Portal [2]. Hourly pedestrian counts at 41 locations in the CBD were captured. Four sensors were found to be frequently non-operational and their data were removed. Missing entries were detected and filled with counts from the previous hour. This accounts for only 0.13% of the total data and does not affect the results. Note that the City of Melbourne is generally unique in making this type of pedestrian count data publicly available. Consequently, our study is focused on POIs from the City of Melbourne.

We combined the Flickr trip data with pedestrian traffic data and performed further filtering to keep trips within 200 meters of the pedestrian sensors. This reduces the dataset to 2586 tours containing 9123 visits to 72 POIs. The pedestrian counts at the POIs were estimated from the mean of the three nearest pedestrian sensors. It should be noted that the counts were street foot traffic, which could partially correlate with the actual number of visitors at the POIs. Due to the challenge of obtaining exact POI visitor counts, we proposed this alternative method of obtaining an estimated POI visitor count. Nevertheless, an estimation of the street foot traffic can still provide travellers with some degree of guidance to infer the actual crowdedness of the venues. Furthermore, this system can be readily deployed in venues where exact counts are available, such as theme parks and museums.

We took a 50-50 split approach for training and testing, i.e., for each user, the first 50% of the trips are used as the training set and the remaining trips as the test set. For each POI, the visiting time was set to 1 hour, as in related previous work [156]. Travel time between two POIs were estimated using their Euclidean distance and a walking speed of 4km/hour, which is also from the literature [87]. Although more accurate travel time estimation models are available, it is not the focus of our study and can be decoupled from our problem.

3.5.2 Benchmark Algorithms

Since our work is the first to consider the crowdedness criterion, we could not find a benchmark algorithm that performs the same task. Nevertheless, we implemented six benchmarks that have been reported in the literature [87][156].

Random (RD): This algorithm randomly selects an unvisited POI as the next POI.

5-Nearest Neighbour (5NN): This algorithm finds the five nearest unvisited POIs and chooses the most popular as the next POI.

10-Nearest Neighbour (10NN): This algorithm finds the ten nearest unvisited POIs and chooses the most popular as the next POI. The reason for two nearest neighbour algorithms is to investigate the impact of the search range.

Iterative Heuristic Approximation (IHA): This is an iterative heuristic search algorithm proposed in [156] which was adapted to our problem. In each iteration, a trip is found by inserting POIs between the origin and destination. The POI inserted must satisfy all the constraints shown in Section 3.3.1. POIs are ranked and selected by computing the ratio of squared profit and cost: $s(i,j) = \frac{Prof(j)^2}{Cost(i,j)}$, where $Prof = Pop$ (Equation 3.2) + Int (Section 3.4.2) and $Cost(i,j)$ is the sum of travel time from POI i to j and stay time at j (Equation 3.1). After each iteration, the trip is recorded and after all iterations have finished, the best trip is selected as the final output.

Greedy algorithm (GD): This algorithm selects the most popular unvisited POI as the next POI.

Integer Programming (IP): This is an integer programming based optimisation algorithm proposed in [87] which finds the optimal solution of an orienteering problem. Since it could not solve the time-dependent orienteering problem, only the popularity information was used.

3.5.3 Variants of PersCT

In addition to the above benchmarks, we also evaluate four variants of the proposed PersCT algorithm.

Vanilla: This variant purely maximizes POI popularity without considering crowdedness or personalization.

Crowdedness-aware (CR): This variant maximizes a combined objective of POI popularity and crowdedness.

Personalized (CF): This variant uses the user-based collaborative filtering to perform personalization and maximizes a combined objective of POI popularity and user interest.

PersCT (CR+CF): This variant is the proposed algorithm that combines CR and CF.

3.5.4 Evaluation Metrics

We assume a trip is planned before a user leaves the origin and we evaluated the performance of the algorithms after the user reaches the final destination. Each trip is evaluated independently and the results of all trips are averaged together to produce the final results. The following metrics were computed for each trip:

1. **Precision (Pre)**: The proportion of true positives that are also found in the recommended trip itinerary. If S_r is the set of POIs recommended, and S_a is the set of POIs actually visited, then precision $Pre = \frac{S_r \cap S_a}{S_r}$
2. **Recall (Rec)**: The proportion of true positives that are also found in the actual trip itinerary. If S_r is the set of POIs recommended, and S_a is the set of POIs actually visited, then recall $Rec = \frac{S_r \cap S_a}{S_a}$
3. **F Score** (F_1): The harmonic mean of Pre and Rec: $F = 2\frac{Pre \times Rec}{Pre + Rec}$
4. **Crowdedness (Crd)**: The mean crowdedness of the POIs at the time of visit in the itinerary.
5. **Popularity (Pop)**: The sum of the popularity of all POIs in the itinerary.
6. **User Interest (Int)**: The sum of user interest of all POIs in the itinerary for a user.

Metrics 1-3 are selected to evaluate how accurately users are modelled using PersCT vs baselines. Metrics 4-6 evaluate the capability of the algorithms to make the best trade-off between finding interesting places for a user and avoiding the most crowded times.

3.5.5 Results and Discussion

PersCT vs Benchmarks

Table 3.2 shows a comparison of the PersCT algorithm versus the benchmarks. Given the origin and destination of the trip, PersCT was run 50 times with different random seeds and we report the mean of all trips. The parameter settings are $\gamma = 5$, $\alpha = 2$, $\beta = 1$, $\rho = 0.8$ (for a parameter selection analysis see Figure 3.6). For each trip, we stopped the execution of an algorithm if the search time was longer than 10 seconds in order to simulate real-life use cases. As our problem is

Algorithm	Pre	Rec	F_1	Crd	Pop	Int
5NN	0.202	0.259	0.227	0.480	4.134	2.683
10NN	0.194	0.268	0.226	0.525	4.794	2.901
GD	0.220	0.295	0.252	0.533	**5.126**	2.985
RD	0.038±0.024	0.059±0.031	0.046±0.026	**0.401±0.012**	1.848	1.920
IHA	0.202±0.017	0.294±0.020	0.239±0.019	0.508±0.005	4.588	**3.446**
PersCT	**0.239±0.018**	**0.322±0.022**	**0.274±0.020**	0.485±0.007	4.368	3.111
IP	N/A	N/A	N/A	N/A	N/A	N/A

Table 3.2: Comparison of the proposed PersCT algorithm and various benchmark algorithms. The results are averaged over 50 runs with different random seeds. Note that deterministic algorithms (GD, 5NN, 10NN) are not affected by randomness of the solution and therefore do not have standard deviation. The Integer Programming (IP) method did not finish in designated time, and no results are reported.

NP-hard, the integer programming algorithm (IP) [87] failed to find a solution within 10 seconds for all trips tested, and thus we excluded its results. It can be seen that PersCT out-performs all other heuristic algorithms in precision, recall and F score. Interestingly, although GD performs the best in popularity, and IHA wins in user interest, PersCT is the most effective algorithm in finding attractive trips for users. This could be due to the difference in the search strategy. GD only selects the most popular POIs and thus its search space is very limited. IHA searches for POIs that give the highest scores per unit time spent, and thus it also tends to select POIs close-by. In contrast, PersCT explores a much larger solution space due to the use of the trail variable to update trajectories for the ants. Additionally, the distance re-weighting scheme implemented by PersCT also gives a boost in performance, as we show in Table 3.6. For other benchmarks, 5NN and 10NN perform slightly worse than IHA, which is expected since the search space is limited to the nearby POIs. Random performs the worst in $< Pre, Rec, F1 >$, which is also not surprising.

In terms of the crowdedness objective, the random method finds the least crowded POIs with the lowest popularity. The second best algorithm is 5NN, and the third best is PersCT. Despite high accuracy in recommendation, PersCT still manages to find trips that are less congested. The greedy algorithm finds trips with the highest popularity, and interestingly they are also the most crowded. 10NN and IHA perform better in crowdedness than Greedy but underperform PersCT in $< Pre, Rec, F1 >$. These results indicate that crowdedness is highly correlated with popularity, and we found the Pearson Correlation score between popularity and crowdedness values of Table 3.2 to be 0.982. This justifies an intuitive belief that more popular places are more crowded, despite

the fact that the data are collected by two completely different systems (pedestrian sensors vs user-uploaded photos). PersCT shows that the crowdedness of trips can be reduced by avoiding visiting the most popular locations at their busiest times to maximize user satisfaction.

Since PersCT is a stochastic algorithm while the benchmarks are deterministic, we evaluated the statistical significance of the results. Table 3.3 shows a summary of the sample distribution of F_1 scores generated by 50 runs of the algorithm with the sample mean (0.274) and the 95% confidence interval (0.269 - 0.281). It can be seen that none of the benchmarks in Table 3.2 are within the 95% range of the sample mean. Moreover, in Table 3.4, we evaluated the P-values for the F_1 scores of the benchmark algorithms with PersCT using the one-sample t-test, which is a classical test to measure statistical significance [44]. All values are significantly less than 0.05, which indicates that the improvement in F_1 score of PersCT over the benchmark algorithms is statistically significant.

Number of runs	95% Lower	Sample Mean	95% Upper
50	0.269	0.274	0.281

Table 3.3: Distributional information of F_1 scores of the proposed algorithm.

	Greedy	Random	5NN	10NN	IHA
P-value	2.1e-10	2.2e-16	2.2e-16	2.2e-16	7.8e-13

Table 3.4: One sample t-test results of F_1 scores of the benchmarks against the proposed PersCT method.

Variants of PersCT

Table 3.5 compares different variations of the PersCT algorithm. Vanilla relies solely on popularity to generate itinerary recommendations, and consequently the results are more similar to GD in Table 3.2. However, Vanilla still out-performs GD by having a higher F_1 score and less crowded trips. When the crowdedness information is combined with Vanilla (the CR variant), less crowded locations are found, despite lower precision and recall values. This is expected since popularity is highly correlated with the crowds. When CF is used with Vanilla, the best precision and recall scores are observed, consolidating our previous observation about personalization. Meanwhile, trip crowdedness is also high for these itineraries, suggesting that the POIs favoured by the users

Variant	Pre	Rec	F_1	Crd	Pop	Int
Vanilla	0.220±0.019	0.306±0.022	0.256±0.021	0.526±0.006	**4.973**	3.018
CR	0.206±0.018	0.285±0.025	0.240±0.021	0.492±0.010	4.842	3.010
CF	**0.241±0.020**	**0.330±0.023**	**0.279±0.021**	0.523±0.005	4.567	**3.148**
CR + CF	0.239±0.018	0.322±0.022	0.274±0.020	**0.485±0.007**	4.368	3.111

Table 3.5: Comparison of variants of PersCT. Vanilla: No crowdedness or collaborative filtering. CR: only crowdedness. CF: only collaborative filtering. CR+CF: both crowdedness and collaborative filtering.

Variant	Pre	Rec	F_1	Crd	Pop	Int
No f_{dist}	0.209±0.017	0.284±0.021	0.241±0.021	**0.480±0.006**	**4.391**	**3.131**
With f_{dist}	**0.239±0.018**	**0.322±0.022**	**0.274±0.020**	0.485±0.007	4.368	3.111

Table 3.6: Comparison of variants of PersCT. No f_{dist}: does not use distance re-weighting. With f_{dist}: includes distance re-weighting.

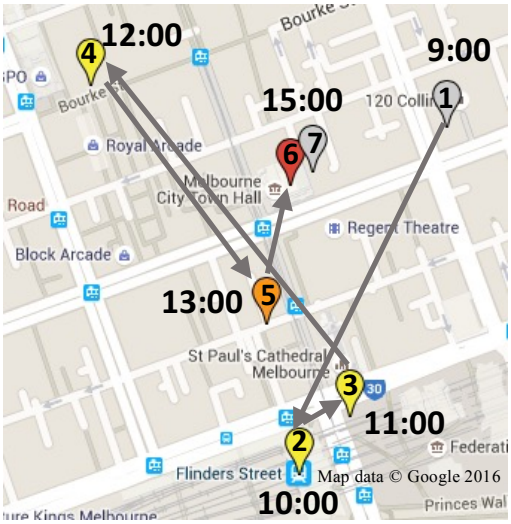
are also popular. Combining crowdedness and personalization, a slight reduction in user interest is observed. However, it is compensated by a large reduction in the level of congestion at the POIs, which can be more desirable for users.

Table 3.6 compares variants of PersCT that include or exclude the distance re-weighting term (Equation 3.9). A significant increase in Precision, Recall and F_1 score and a slight reduction in popularity and user interest can been observed when the re-weighting scheme is applied. This could suggest that for some trips, the most popular or interesting POIs are far away from the user. However, real users tend to prefer POIs that are close to the final destination or at least in the same direction of travel. The results have shown the effectiveness of PersCT and the importance of using geo-graphical information in trip recommendation.

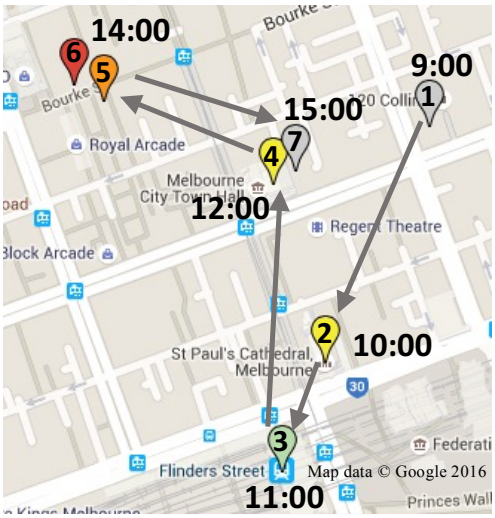
Case Study

We illustrate the effectiveness of the proposed PersCT algorithm with a real life case study. In Figure 3.4, we compare three algorithms, namely PersCT, greedy and the CF variant, to recommend a trip for a user who starts at 9:00 and has a time limit of 7 hours. The colors of the POIs indicate its crowdedness at the time of the visit. The first observation is that most POIs are more crowded in the afternoon than the morning, which is expected. It can be seen that the PersCT algorithm plans the trip such that POIs that are more crowded in the afternoon are visited in the

(a) PersCT



(b) Greedy



(c) CF

Color	Crowdedness Value
	$0.8 < Crd \leq 1$
	$0.6 < Crd \leq 0.8$
	$0.4 < Crd \leq 0.6$
	$Crd \leq 0.4$
	None

(d) Color Legend

Figure 3.4: A case study showing the trips planned using (a) PersCT, crowdedness = 0.478 (b) Greedy method, crowdedness = 0.563 (c) PersCT without crowdedness objective (CF only variant), crowdedness = 0.598.

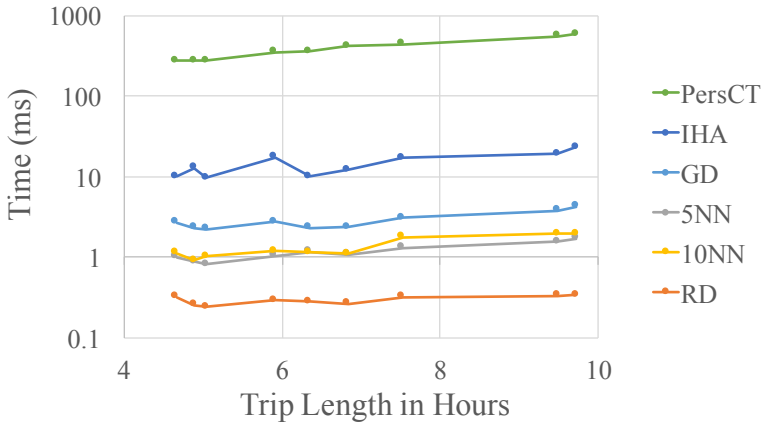
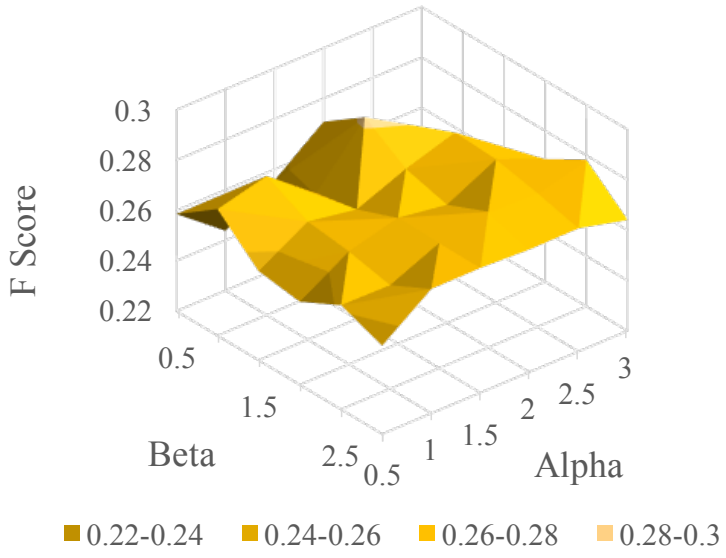


Figure 3.5: Running time in milliseconds.

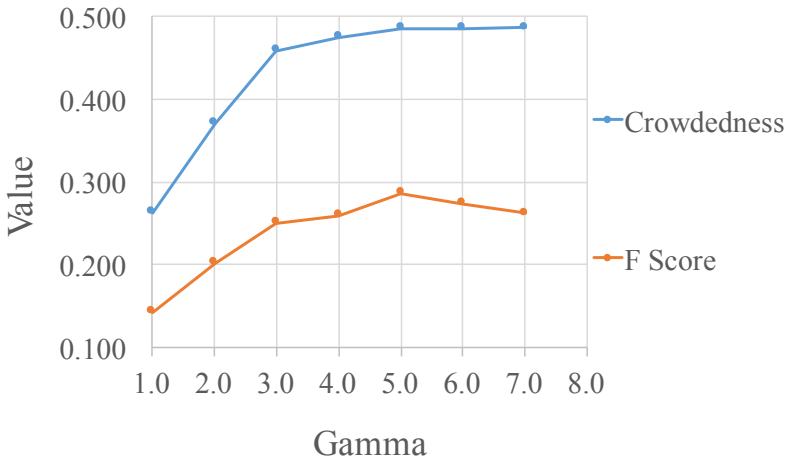
morning instead (POI 2 and 3, which are in a busy shopping region) and POIs less sensitive to temporal foot traffic variations are scheduled in the afternoon (Figure 3.4(a)). All POIs visited have crowdedness values of less than 0.8, which should be acceptable to most users. For the Greedy method 3.4(b), since the most popular locations are selected first, the distance travelled for the trip is far from optimized. Moreover, it is not surprising that the crowdedness is high as a consequence of recommending popular POIs first. POI 7, which is very congested in the afternoon, significantly increases the overall crowdedness and it is avoided in the trip recommended by PersCT. A second example, Figure 3.4(c) shows the resulting trip if only the collaborative filter is used in PersCT. The crowdedness of the trip is even higher than the greedy solution. This is also expected as the algorithm has no information about the crowd profile at each POI.

Running Time

We implemented PersCT in Java7 and performed the evaluation on a 2.7GHz Macbook Pro with 8GB of RAM. We set the number of iterations to be 20 as no significant improvement in performance was observed for higher values. The running time of PersCT is compared with various benchmark algorithms in Figure 3.5. The running time of all evaluated algorithms increase with the trip length, which is expected due to the growing number of POIs that must be ordered. As the time complexity of the Ant Colony meta-heuristic is quadratic in the number of nodes, PersCT is slower than the benchmarks. However, the longest running time recorded was at trip length



(a) F_1 vs α and β



(b) F_1 vs γ

Figure 3.6: Performance of the PersCT algorithms with varying parameter settings. (a) Changing α and β. (b) Changing γ.

= 9 hours and it is still less than one second in Java without any code optimisation. Given the improvements of PersCT over the benchmarks in terms of Pre, Rec, F1 and Int, we consider that the small trade-off in running time is acceptable as it is sufficient for most real-life applications.

Parameter Selection

We performed a grid search for the two important parameters, α and β, of the PersCT algorithm (see Section 3.4). These two values determine the preference of searching for new POIs (exploration) versus trying a different combination of currently selected POIs (exploitation). If $\alpha < \beta$, the algorithm is tuned towards exploration, and vice versa. The results are shown in Figure 3.6(a). Each experiment was run 10 times and the results were averaged. The best performance was observed at $\alpha = 2$ and $\beta = 1$. This is reasonable as a large α is likely to give a locally optimal solution and a large β will result in under-optimised solution. We also evaluated the impact of changing γ, which determines the balance between maximizing the profit function and minimizing crowdedness. A high γ will instruct PersCT to search for POIs with higher popularity and user interest level, whereas a lower value will results in less congested venues to be selected. As can be observed in Figure 3.6(b), the highest F score can be achieved when γ was set to 5.0, although the results were largely consistent over the range $\gamma \in [3.0, 7.0]$

3.6 Conclusions and Future Work

In this chapter, we studied trip recommendation using trajectories and smart sensing. We formulated the personalized crowd-aware trip recommendation problem based on the Orienteering Problem and modelled our problem as a multi-objective time-dependent OP with time-dependent objectives. Given user travel histories and foot traffic information collected by sensors, we proposed the PersCT algorithm, which extends the Ant Colony Optimisation meta-heuristic and extracts POI popularity, user interest and crowdedness information to recommend low congestion trips that also suit the users. We evaluated PersCT using real life datasets from Flickr geo-tagged photos and six months of pedestrian traffic data in Melbourne, and we showed that the proposed PersCT algorithm out-performs a number of benchmark algorithms in precision, recall and F_1 score, as well as achieving a good balance in crowdedness.

We acknowledge the following limitations of this study and propose some directions for future research:

1. Due to data availability issues, we were unable to obtain pedestrian count datasets in other cities to further evaluate our algorithm. A potential alternative option is to use pedestrian simulation models and evaluate accordingly. However, some calibration data are still needed in many simulation models. Obtaining a reliable pedestrian model could be the next step. Other data sources such as mobile phone logs could also be explored.

2. We have assumed that the number of users of this system is a small fraction of the overall population and does not affect the pedestrian distribution at the POIs. In real-life situations where a POI has limited availability, this might not be true. Future work could be to investigate recommending trips for multiple tourists using the system simultaneously, and a load balancing scheme may be the next step in this direction.

3. We used the weekly average historical pedestrian distributions to estimate the daily pedestrian volume since the dates of the visits and pedestrian sensor measurements do not match. For a real-life implementation, we would have obtained the data on the same day and we could estimate crowdedness based on real-time data, which can provide more accurate estimate of the crowdedness.

4. Each user potentially has a different level of tolerance for POI crowdedness. Due to the lack of data, we did not estimate personalised crowdedness tolerance in this work and it could be a direction for future study.

5. In this work, we infer temporal information of the POIs from pedestrian volume data, and we have assumed that user interest is independent from time. An alternative approach would be to extract temporal information from geo-temporal tagged photos. However, due to the sparsity of temporal tagging, inferring meaningful temporal information is very difficult. For example, it is common for two consecutive POIs visited by the same user to have a time difference greater than 8 hours. We have experimented with the approach suggested in [87] to infer temporal user interest. However, it did not perform well on our dataset. Making full use of geo-temporal tagged photos could be a future research direction.

6. Although the opening hours of the POIs are relevant to most users, they are not considered in this study due to the lack of an efficient method to obtain such data for every POI. If such information is available in the future, it is equivalent to adding an additional constraint such that the time of visiting each POI must be within its opening hours. We could adapt the PersCT algorithm such that when an ant arrives at a POI during the search, it compares the current time with the opening hours and terminates the search when the constraint is violated. As the PersCT algorithm uses a bottom-up approach, the addition of constraints can easily be accounted for. The only adverse effect would be that some solutions in an iteration would become invalid, and consequently more iterations may be needed.

7. As shown in Figure 3.5, although the proposed PersCT algorithm out-performed the benchmarks, it is computationally more expensive. To reduce the running time, two obvious choices would be (1) reducing the number of iterations performed, and (2) reducing the number of ants used in each iteration. However, they would inevitably have an impact on the performance of the algorithm as both choices decrease the amount of search executed in finding the solution. A third direction could be to cache the states of pedestrian volumes for given origin-destination pairs for future queries. When a new query arrives, we could first search inside the cache for similar origin-destination pairs and pedestrian volume states. Whenever there is a cache-hit, the amount of search required can be significantly reduced, and the running time would be faster. For a cache-miss, however, the running time would increase when compared to the non-cached PersCT algorithm. Designing an appropriate strategy to best utilise the limited amount of memory by maximising the cache hit rate would be an important problem for future research.

Chapter 4

Road Traffic Analysis Using Contrast Mining on Vehicle Trajectories

There is growing interest in using trajectory data of moving vehicles to analyze urban traffic and improve city planning. This chapter presents a framework to assess the impact of traffic intervention measures, such as road closures, on the traffic network. Connected road segments with significantly different traffic levels before and after the intervention are discovered by computing the Growth Rate of their traffic. Frequent sub-networks of the overall traffic network are then discovered to reveal the region that is most affected. The effectiveness and robustness of this framework are shown by three experiments using real taxi trajectories as well as traffic simulations in two different cities. The publication arising from the work in this chapter is paper P2.

4.1 Introduction

TRAFFIC monitoring and control is a critical problem in modern cities. Despite the recent introduction of advanced technologies such as Intelligent Transportation Systems (ITS), traffic congestion remains a major challenge to urban designers and transport authorities. In Melbourne, the worsening of traffic congestion is reflected by the reduction of average travel speed by around 4km/hr in the metropolitan area during the period 2005 - 2014 [138]. The situation is more severe in "mega cities" like Beijing. One of the most notable events was a 100-km traffic jam in August 2010 on a major highway (National Highway 110) heading to Beijing, which has been reported as the "World's Worst Traffic Jam" [51]. The highway is a main corridor for trucks transporting coal to Beijing and nearby provinces, and August is the peak season of the year. In 2010, construction work began in mid-August on the highway, which reduced the road capacity by 50% and caused the severe congestion [123]. Despite swift actions taken by the authorities, the traffic

jam still took around 10 days to clear, which resulted in significant economic losses. Although this example is relatively extreme, it is not uncommon to find oneself stuck in a traffic jam due to road work. Therefore, the monitoring and characterization of the impact of traffic events, such as road closures, is an important and challenging problem in urban traffic management. Current ITS technologies usually employ induction loop sensors or street cameras at traffic intersections to monitor the traffic condition [74], which can provide helpful information to the authorities. However, sensor coverage is usually low compared to the coverage of the road network, which results in the lack of data granularity. Moreover, these sensors are unable to identify the overall trajectories or paths of the vehicles, and the monitoring of the change in the traffic flow as a result of external events can be difficult.

Advances in ubiquitous sensing technology have provided an alternative solution. For example, taxi companies have been equipping vehicles with GPS sensors, and each vehicle can be seen as a small sensor that generates trajectory information. With thousands of vehicles operating daily, their trajectories can be used as a source of traffic flow information to monitor road traffic, which nicely complements data gathered by induction loop sensors. Indeed, several papers have proposed the use of GPS sensor data to identify traffic anomalies or congestion caused by urban design problems [91, 107, 109, 147]. However, characterizing changes in traffic flows under the impact of road closure events has received comparably little attention.

In this chapter, we propose a framework to analyse changes in traffic flows due to road closure events based on GPS trajectory data. Specifically, we employ ideas from contrast mining and frequent itemset mining to define, characterize and visualize the changes. Our major contributions are:

1. *Traffic Modelling.* We model the traffic network of a city as a graph G, extract nodes and edges and map GPS trajectories to the graph. We then search for traffic flow sequences that frequently occur and generate *n-Edgesets*, which represent frequent sets of edges.

2. *Mining Emerging n-Edgesets.* We partition the data into subsets that are captured before and after the road closure event and compute the *Growth Rate* to characterize *n-Edgesets* that have exhibited changes in flow. We then use the Local Outlier Factor (LOF) algorithm to identify *Emerging n-Edgesets*, i.e., sequences of road segments with significantly increased or decreased traffic.

3. *Mining Frequent Emerging Network.* We propose the MineFreqNetwork algorithm to find frequently occurring Emerging n-Edgesets to form a coherent network. This Frequent Emerging Network reveals the most severely affected sub-network of the city as a result of the road closure.

The rest of the chapter is organized as follows. Section 4.2 introduces relevant background information to the problem, and reviews related work. Section 4.3 defines the problem and gives an overview of our framework. In Section 4.4, we discuss our methodology in detail. Section 4.5 presents the experiments and evaluation of our algorithms. In Section 4.6, we conclude and discuss future research directions.

4.2 Background and Related Work

In this section, we review related work in contrast mining and trajectory mining with a focus on road traffic analysis. Some background knowledge is also introduced when necessary.

4.2.1 Trajectory Mining in Road Traffic Analysis

Trajectory mining in road traffic analysis has received considerable attention recently as more data are becoming available from on-board GPS sensors in taxis. The trajectories are usually acquired as a time series containing the latitude and longitude coordinates of the vehicle, the timestamp of the GPS signal, the id of the vehicle and sometimes the occupancy status of the vehicle if it is a taxi. Trajectory mining has been used in a variety of traffic analysis problems, including traffic modelling and prediction [24], travel time estimation [154], destination prediction [143] and event detection [159]. Zheng et al. [161] have given an extensive review on the recent development of trajectory mining in traffic analysis. Although the problem of characterizing flow changes has not received sufficient attention, one research direction closely related to our problem is the detection of anomalous trajectories. Thus, we mainly review these works in this section.

An anomaly, sometimes called an outlier, is “an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism” [65]. From a probability distribution point of view, an anomaly is unlikely to belong to the same distribution as the majority of the data points [3]. A large number of anomaly detection algorithms

have been proposed, and we mainly review the works that focus on detecting trajectory anomalies. Surveys of general-purpose anomaly detection algorithms can be found in [27, 66, 160].

An anomalous trajectory is a trajectory that is very different from the rest of the trajectories. The purpose of detecting anomalous trajectories may vary in different applications, and therefore, the definition of being "anomalous" may change. Applications for anomalous trajectory detection can be classified into two types, *micro* or *macro*. *Micro* applications focus more on the individuals who use the traffic system. An example is the detection of malicious detours by taxi drivers to reap more profits from customers. Liu et al. proposed a speed-based clustering method to detect malicious detours by taxi drivers [90]. The authors identified frequently visited areas for each taxi and computed the average speed for each area. The data were clustered and suspicious driving routes were identified by computing the deviance between the speed of the current trip and the average speed in the active area. Zhang et al. tackled a similar problem using an isolation-based anomaly detection technique [157]. The sparse taxi trajectories were mapped to grids and interpolated with a shortest path method. A binary tree was populated by classifying trajectories into ones that contain or do not contain a random grid, and anomaly scores were calculated by averaging the number of grids used to isolate a trajectory.

Unlike *micro* applications, *macro* applications focus on the system as a whole. An example of anomalous trajectory detection for *macro* applications is the identification of accidents from unusual detouring of vehicles at a road intersection. Pan et al. detected traffic anomalies in taxi GPS data by computing the Mahalanobis distance [38] between the traffic volume across two regions. Two search indices were constructed to speed up the anomaly detection process and frequently occurring terms in online social media were retrieved to describe the anomalies [107]. Pang et al. used a likelihood ratio test to identify regions that have high likelihood of being anomalous [108, 109]. Lee et al. detected outliers in trajectories that were not bounded by grids using a clutering-based approach [83]. Sub-trajectories were first partitioned using the minimum description length (MDL) principle [117] and then further split into finer segments by their angle or distance to the end point.

In addition to detecting anomalous trajectories in traffic, some authors focused on characterizing the anomalies. Zheng et al. [163] modelled city-wide traffic by constructing a connectivity matrix whose entries are the traffic flows captured by trajectories between each region. Features

of the traffic flow such as speed, traffic volume and distance between centroids of region pairs were aggregated into a matrix. The features were then clustered based on the speed and distance features, and three clusters were found, which correspond to (1) low speed, small distance, (2) low speed, large distance, and (3) high speed, large distance. The transition between each cluster of a region in a day was analyzed by mining frequently occurring state transitions between days, and badly connected regions were found. Liu et al. [91] went further by discovering spatio-temporal causal relationships between anomalous traffic links using a tree structure. Frequently occurring traffic anomalies were inserted into the tree and the appearance and disappearance of such anomalies were shown. Chawla et al. [29] mined traffic anomalies using Principle Component Analysis and then inferred the routes that caused the anomalous trajectories with an L1 optimisation technique.

These works mainly focus on discovering traffic anomalies to identify urban design issues, environmental problems or regulate taxi driver behaviour. In contrast, our work addresses a different question: once some traffic intervention occurs, for instance, building a new road or closing a road segment, what will the impact be of such an event on the traffic flow? To the best of our knowledge, this problem has not been given sufficient attention. Miller and Chetan [97] have studied the impact of short-term highway traffic incidents. In our framework, we are not restricted to highways and also consider long-term traffic interventions that can last for weeks or months. Salcedo-Sanz et al. [119] performed simulations to reconfigure one-way streets in a town to find the shortest path in order to cross an area affected by a large event. However, real traffic data were not used, whereas we use both real vehicle GPS data and simulated vehicle trajectories to identify regions of impact. In transport systems research, loop sensors that are embedded under the road network are typically used to monitor traffic. Banaei et al. [12] analyzed loop sensor data in Los Angeles and clustered road segments to find a limited number of distinctive signature traffic patterns on all road segments. However, the connections between road segments based on the traffic flows are not discussed, which is a major limitation in using loop sensors. Our approach can be configured to study a single road segment or a connected set of roads, which can provide traffic information that is not available from loop sensors.

In addition to detecting and characterising trajectory anomalies, some authors have used trajectory mining to estimate travel time or speed. Wang et al. proposed a two-step framework to

estimate travel time using GPS trajectories [140]. The authors partition a day into 30-minutes time slots and use the GPS trajectories captured by taxis in each time slot to provide real-time estimates. The travel time of a path was computed by finding the optimal sequence of trajectories that minimize the error between the estimated travel time and historical travel time generated from frequent sub-paths. Shang et al. proposed a matrix-factorisation model to estimate travel speed [121]. An intrinsic inverse relationship between speed and volume was observed and the authors then used a partially observed Bayesian Network to infer unknown traffic volumes on small roads. These works have not considered the impact of special events on the estimates, and therefore an event may generate large estimation errors. Although in this work, we do not consider travel time estimation, our work may be useful to improve these models, and we discuss the future research directions in Section 4.6.

4.2.2 Contrast Mining

Contrast mining is a field in data mining that focuses on identifying the differences between or among two or more datasets [46]. Most often, the datasets being contrasted are formed by splitting a single data source by criteria such as time, spatial location or class [46]. Interesting patterns can often be discovered when one dataset is compared against another. These patterns may be helpful to domain experts for further analysis or building a classifier. For instance, a combination of features that differentiate edible and poisonous mushrooms were found in the Mushroom Data from the UCI machine learning repository [86]. Pattern X = {(odour = none), (gill size = broad), (ring number = one)}, and Pattern Y = {(bruises = no), (gill spacing = close), (veil colour = white)} [47]. Pattern X is present in 63.9% of edible mushrooms and 0% of poisonous ones, whereas Pattern Y is observed in merely 3.8% of edible mushrooms but 81.4% of poisonous ones [47]. Using these patterns, a classifier that accurately predicts whether a mushroom is edible or poisonous can be designed [47].

The idea of contrast mining has been used to analyze a large number of datasets in various domains. In this section, we survey the contrast mining literature and summarize the proposed algorithms and their applications. Dong and Bailey are the pioneers in this field and their book on contrast mining can be found in [46].

To perform contrast mining, one must first define the *contrast pattern* that is being mined. The

two most common definitions are *Growth Rate* [47] and *support delta* [15]. The *Growth Rate* of a pattern X for a dataset D_j when compared with dataset D_i is defined as

$$gr(X,D_j) = \frac{supp(X,D_j)}{supp(X,D_i)} \tag{4.1}$$

where $supp(X,D_i)$ is the support of pattern X in dataset D_i [46]. Support is a concept that originated from association rule mining in transaction data, and $supp(X,D_i)$ is defined as the proportion of the pattern X occurring in all the patterns of the database D_i [155]. For example, if D_i contains 100 strings that consist of the letters A, C, T and G, and 10 of the strings contain the pattern $X = ATGCCTAG$, then $supp(X,D_i)$ is 10/100 = 0.1. If in a different dataset D_j, the support of X is 0.2, the *Growth Rate* $gr(X,D_j)$ when contrasted with D_i is 0.2/0.1 = 2. Another common *contrast pattern* is the *support delta*, which is defined as

$$supp_\delta(X,D_j) = supp(X,D_j) - supp(X,D_i) \tag{4.2}$$

between two datasets D_j and D_i. As *Growth Rate* and *support delta* work similarly, we mainly review the literature that focuses on the former.

Contrast patterns that characterize the *Growth Rate* of a dataset are also called *emerging patterns* (EP) [46]. In the example above, the pattern X is said to be a ρ-*emerging pattern* (ρ-*EP*) when $gr(X,D_j)$ is larger than a threshold ρ. Different algorithms have been proposed to mine *emerging patterns* for different datasets, including multi-dimensional vectors, spatial and temporal data. In the next section, we summarize work for these three data types and make a comparison with our problem of mining contrast patterns for vehicle trajectory data.

Mining Multi-dimensional Vectors

Multi-dimensional vector data were the first data type that were investigated in the contrast mining community. As *emerging patterns* characterize abrupt changes in the support of data, researchers have been interested in the discriminative power of EPs in the design of classifiers. One of the earliest works is [47], where the authors mined *emerging patterns* in the US census data [135] and UCI Mushroom dataset [86].

To extract discriminative sets of features using EPs, a naive algorithm requires the computation

of all combinations of features, which is prohibitive for a high dimensional dataset such as the US census with 75 attributes. Moreover, the apriori principle does not hold for EP computation [47], which increases the difficulty of pruning irrelevant feature sets. Apriori is an algorithm proposed by [4], which was initially used for frequent itemset mining. Consider an attribute of a vector as an *item*, and a set of attributes as an itemset, then frequent itemset mining aims to find itemsets in a database that have a support value larger than a certain threshold. The apriori algorithm effectively prunes itemsets using a bottom-up approach: starting from itemsets that contain a single item, enumerate all possible itemsets as the number of items increase. If a certain itemset A has a support value below the threshold, prune all other itemsets that contain A as they must all be no more frequent than A. This approach can significantly reduce the number of itemsets enumerated. However, for EPs, if the *Growth Rate* of an itemset A is below the *Growth Rate* threshold, it does not necessarily mean that all itemsets containing A will also have low *Growth Rate*.

To address this problem, the authors of [47] proposed an algorithm that exploited *borders*, which are special sets that reduce the number of itemsets to be generated. The Max-Miner algorithm [16] was employed along side with *borders* to achieve the search speed-up. Despite the reduction in the number of EP candidates using *borders*, the worst case time complexity was still exponential. Bailey et al. focused on mining *jumping emerging patterns* (JEPs), a special type of EP whose support abruptly increases from zero in one dataset to non-zero in another [11]. This property was shown to have discriminative power in classification [11]. A tree structure was proposed for the mining of JEPs, which achieved a speedup of up to 10 times compared with Max-Miner. Fan et al. introduced *strong jumping emerging patterns* (SJEPs), which are special JEPs with an additional minimum support threshold, and proposed a *CPtree* structure to efficiently mine SJEPs [49]. SJEPs were further used to construct a classifier, which achieved excellent results against several benchmarks [49].

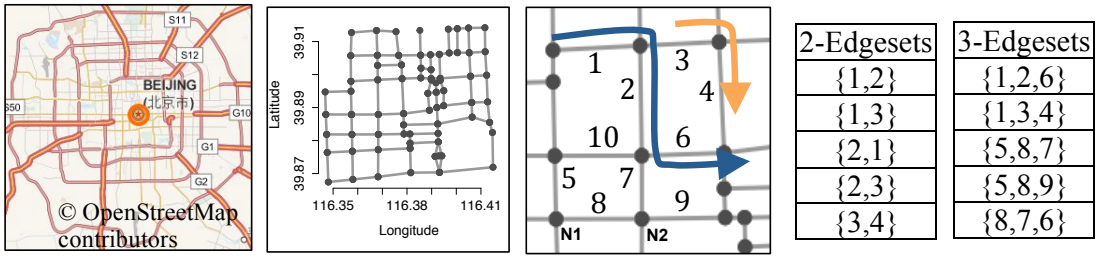
In this chapter, we are interested in mining emerging patterns in vehicle trajectory data, which is a more complex data type than static vectors discussed above, since (1) trajectories contain both spatial and temporal patterns, (2) trajectories have uncertainty due to GPS sampling rate and drift issues, and (3) the volume of trajectories can greatly exceed previous datasets, which can pose new challenges in computational cost.

Mining Spatial and Temporal Data

Despite being widely used in the classification of static vector data, *emerging pattern* mining has not received as much attention in geo-spatial applications or time series data mining. Spatial data typically consist of lines, points, polygons and arcs that are used to build digital maps. These objects are usually stored in a spatial database, which is a special relational database with support for queries targeting spatial data such as nearest neighbour queries. Ceci et al. [25] was the first to apply EP mining to spatial databases by means of SQL queries, although spatial neighbouring relationships were not considered. Takizawa et al. [129] analyzed room layouts in the Japanese rental market and developed a graph-based model using EPs to identify causes of high or low rental prices. The author further investigated crime spot distributions and used EPs to predict likely occurrences of crimes [130]. Ding et al. [43] extracted EPs in satellite images of vegetation coverage in the USA. The authors proposed an iterative algorithm to identify the optimal boundary between high and low vegetation areas. Each image pixel is randomly initialized to either a high state or a low state representing the vegetation density. The transitions between the two states are defined by a Markovian transition model, and the transition probability is affected by the state of its neighbouring pixels. Scores are rewarded when a pixel transition generates EPs with higher *Growth Rate*, and the system is run until convergence. A similar spatial analysis was performed in [126] where the authors extracted discriminative patterns from the 2008 US election and showed correlation between social-economical features of a region and electoral support. However, unlike static spatial data, the neighbouring relationship of vehicle trajectories is difficult to define as a vehicle is free to move on the map and two consecutive trajectory points may be on different roads.

4.2.3 Summary

Contrast mining (or *emerging pattern*, EP mining) has been extensively studied for data types such as multi-dimensional vectors, and the majority of the papers studying EP mining has focused on designing more accurate classifiers that can produce results interpretable to humans. Some work has also been dedicated to mining EPs for spatial data processing. However, the unique challenges brought by trajectory data, such as sparsity caused by sampling rate, render the current



2-Edgesets	3-Edgesets
{1,2}	{1,2,6}
{1,3}	{1,3,4}
{2,1}	{5,8,7}
{2,3}	{5,8,9}
{3,4}	{8,7,6}

(a) A road network (b) A Road Network Graph (c) Vehicle Trajectories (d) n-Edgesets, n = 2 and 3

Figure 4.1: An example illustrating the proposed contrast mining framework. (a) Road network of Beijing. (b) Graph model extracted from the road network. (c) GPS trajectories showing traffic flow. (d) Edgesets representing directions of traffic flow.

contrast mining algorithms unsuitable for trajectory analysis. Trajectory mining has been a popular research field in recent years with many papers studying anomalous trajectory detection or travel time prediction. However, to the best of our knowledge, using contrast mining to characterize changes in traffic flows based on trajectory data has not been studied before.

4.3 Preliminaries and Problem Statement

In this section, we introduce several definitions and formally define our problem of emerging sub-trajectory mining.

4.3.1 Preliminaries

Definition 4.1 Road Network Graph. A Road Network Graph G (Figure 4.1 (b)) is an undirected graph where road intersections form its nodes and road segments form its edges. Each node in G is a triplet $\langle$NodeID, longitude, latitude$\rangle$ and each edge is a triplet $\langle$EdgeID, N_1, $N_2\rangle$ where N_1 and N_2 are the node pair connecting the edge.

Definition 4.2 Trajectory. A trajectory, *Traj*, is a sequence of time-stamped geographical locations of a moving object. A *Traj* can be expressed as a vector of triplets $\langle$longitude, latitude, time$\rangle$. Figure 4.1 (c) shows two trajectories.

Definition 4.3 n-Edgeset. An n-Edgeset *nES* is an itemset consisting of a sequence of *n*

connected edges in a Road Network Graph. An *nES* represents the direction of traffic flow though the edges. Figure 4.1 (d) illustrates some 2-Edgesets and 3-Edgesets. Traffic flow from edge 1 to edge 2 is denoted as $\{1,2\}$. The trajectories in Figure 4.1 (c) traverse 3-Edgeset $\{1,2,6\}$ and 2-Edgeset $\{3,4\}$.

Definition 4.4 Emerging n-Edgesets. Given two time periods T_1 and T_2, if the traffic profile of an n-Edgeset changes significantly during these times, it is called an Emerging n-Edgeset (*EnES*).

Definition 4.5 Frequent Emerging Network (*FEN*). A connected sub-network of a Road Network Graph is a Frequent Emerging Network (*FEN*) if it consists of frequently occurring Emerging n-Edgesets.

4.3.2 Problem Statement

Given road map data *R* and vehicle trajectories *Traj*, construct a Road Network Graph *G* and find all Emerging n-Edgesets (*EnES*) in *Traj* before and after an event *V*. Identify the Frequent Emerging Network (*FEN*) using the *EnES*.

4.4 Our Approach to Mining Frequent Networks

In this section, we first give an overview of the framework before presenting our method in three sub-sections: (1) traffic network modeling (2) detecting Emerging n-Edgesets (*EnES*) and (3) detecting the Frequent Emerging Network (*FEN*).

4.4.1 Overview

Figure 4.2 shows an overview of our framework for mining frequent emerging networks from trajectory data. The road network is preprocessed by extracting road segments and intersections from raw map data. Traffic flows are modelled by mapping GPS trajectories to the roads (Section 4.4.2). The processed trajectories are scanned to search for *n-Edgesets* and to calculate the *Growth Rate* of traffic after an event has occurred. Road segments that have considerable change in the traffic flow are found by identifying *Emerging n-Edgesets* (Section 4.4.3). Finally, the parts of the traffic network that frequently emerge as the areas with *Emerging n-Edgesets* are found (Section

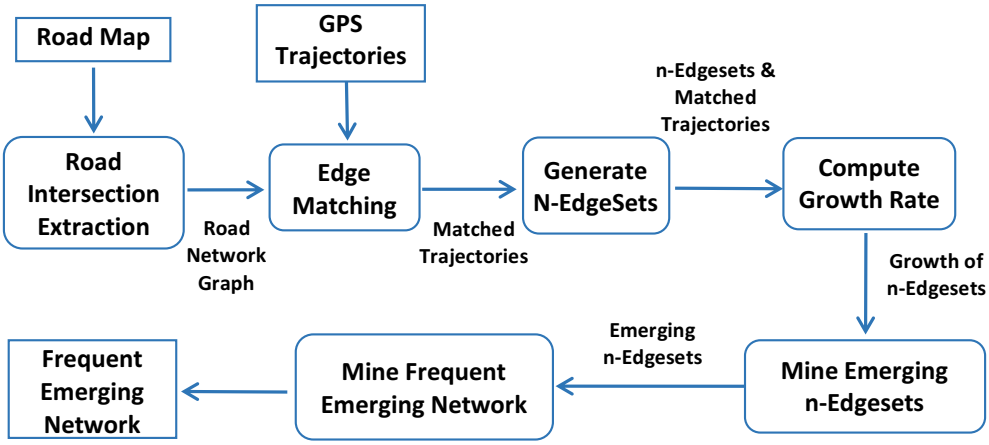


Figure 4.2: Framework of our method.

4.4.4).

4.4.2 Traffic Network Modelling

We model a road network as an undirected graph where intersections are the nodes and road segments are the edges. Since direction information is embedded in the trajectory data, the road network does not have to be directed, which simplifies the computation. This modelling approach is different from some region-based approaches such as dividing the map using regions enclosed by major roads [146] or rectangular grids. Region-based approaches can be useful in providing higher-level semantic information, but the capability of inferring traffic conditions on individual roads is limited. In contrast, modelling road segments can provide finer details about the traffic network as well as capturing more accurate trajectory information. We extracted road segments and intersections manually from OpenStreetMap [106] to ensure the nodes and edges are correctly connected. Although this approach is time-consuming, a properly modelled road network is highly important. For larger scale applications, the accuracy of the road network may not be as significant since high level information can be sufficiently informative, and therefore region-based approaches may be used.

We focus on finding the *Growth Rate* of the traffic on n-Edgesets, which gives information about the change in traffic flow following an event.

3-Edgesets	TraffBe	SuppBe	TraffAf	SuppAf	Growth Rate
{1,2,6}	9	0.09	0	0.00	0.00
{1,3,4}	10	0.10	16	0.16	1.58
{5,8,7}	19	0.19	16	0.16	0.83
{5,8,9}	30	0.30	31	0.30	1.02
{8,7,6}	33	0.33	39	0.38	1.17

Table 4.1: An example of Emerging 3-Edgesets. TraffBe: Traffic before event. SuppBe: Support of Edgeset before event. TraffAf: Traffic after event. SuppAf: Support of Edgeset after event.

Since raw vehicle trajectories do not carry well-defined neighbouring relationships, we must map the "free" trajectories to a confined graph structure so that sequential information can be extracted. Therefore, the second step in our traffic network modeling is to map GPS trajectories to the edges of our Road Network Graph. This is performed by a spatial nearest neighbor (NN) algorithm. Between any pair of connected nodes, a linear interpolation of dummy nodes is created. These dummy nodes carry the same edge ID as the edge connecting the real nodes. For each trajectory point, we find its nearest dummy node within a certain radius r and the edge ID of that dummy node is used as the edge ID of the point, thereby converting its representation ⟨longitude, latitude, time⟩ to ⟨EdgeID, time⟩ (see Definitions 4.1 and 4.2). We use this approach due to its computational time efficiency (O($n$log(n)) when implemented using a kd-tree [17]).

4.4.3 Mining Emerging n-Edgesets

For each edge e in a Road Network Graph G, we exhaustively search for all connected sets of non-repeating edges of depth n that start from e, and each of these sets forms an n-Edgeset. Table 4.1 illustrates an example of several 3-Edgesets in Figure 4.1 (c). Edgeset {1,2,6} represents the traffic flow in the following path: edge 1 to edge 2 to edge 6. Using the n-Edgesets and the GPS trajectories, we calculate the traffic volume on each n-Edgeset for different time periods. The vehicle trajectory database is divided into two parts, D_1 and D_2, where: D_1 = data collected before a road closure event; D_2 = data collected after the closure (Section 4.5 gives more details). Given the n-Edgesets, at each time step i, n new trajectory points are retrieved from the database. The *id* of these new trajectory points are compared to ensure that they belong to the same car. The edge labels are then matched to the list of n-Edgesets. If a match is detected, the traffic volume count of that n-Edgeset is incremented.

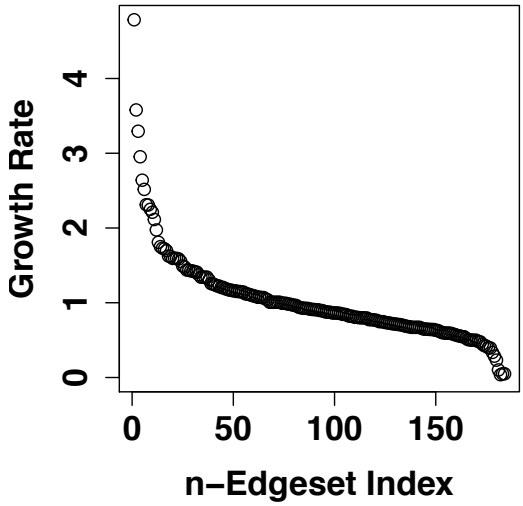
Using the traffic volume on each n-Edgeset, we compute the support of all n-Edgesets in D_1 and D_2 where the support of an n-Edgeset X, $Supp_X(D_1)$, is the traffic volume on X divided by the sum of the traffic volumes on all n-Edgesets in D_1. Assuming that the routing behaviour of most drivers does not change significantly on a daily basis, the difference in the support is likely to carry information about the impact of the event. Therefore, we compute the *Growth Rate* (or simply *Growth*) of the support of an n-Edgeset using the following definition:

Given two datasets D_1 and D_2,

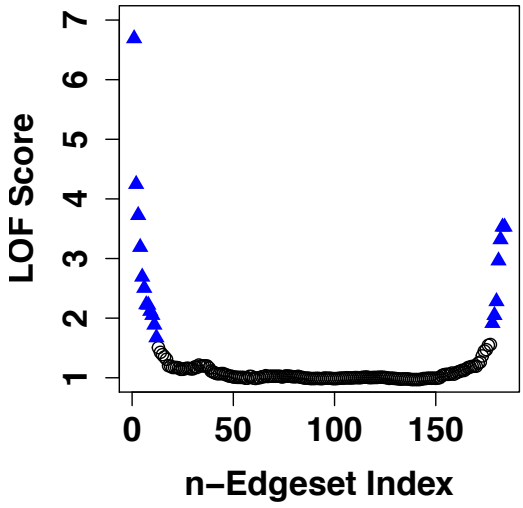
$$GrowthRate_X = \begin{cases} 0, & \text{if } supp_X(D_1) \neq 0 \text{ and } supp_X(D_2) = 0 \\ \infty, & \text{if } supp_X(D_1) = 0 \text{ and } supp_X(D_2) \neq 0 \\ \frac{supp_X(D_2)}{supp_X(D_1)}, & \text{otherwise} \end{cases} \tag{4.3}$$

This concept was initially proposed by Dong and Li [47], but to the best of our knowledge, it has never been used in road traffic analysis. The original definition defines *Growth* = 0 if $Supp_X(D_1) = 0$ and $Supp_X(D_2) = 0$, and therefore, this is also the definition used by us. Table 4.1 shows the *Growth Rate* of several 3-Edgesets. For example, Edgeset $\{1,3,4\}$ has support before the event = 0.10 and after the event = 0.16. Thus its Growth = 0.16/0.10 = 1.58.

After computing the *Growth Rate* of n-Edgesets, we form a rank ordering of the n-Edgesets by sorting them in decreasing order of their corresponding *Growth Rate*. Figure 4.3(a) shows a plot of the sorted *Growth Rate* for 2-Edgesets between consecutive days. It can be seen that the distribution consists of three regions, two tail regions and one flat region. The majority of the 2-Edgesets have *Growth Rates* of approximately 1.0, which forms the flat region in the middle of the plot. This is expected since the traffic volume on most major roads is unlikely to change significantly and the *Growth Rates* are mostly attributed to random noise. *Growth Rates* of the 2-Edgesets at the tails of the sorted edgesets deviates significantly from the flat region. These 2-Edgesets are most likely under the effect of the road closure and may contain interesting patterns to reveal the relationships between traffic flows. Therefore, the next problem is to reliably identify the 2-Edgesets that are in the tail regions.



(a) Growth Rate



(b) LOF Scores

Figure 4.3: Growth Rate and LOF scores. (a) Growth Rate for 2-Edgesets. (b) LOF scores with number of neighbors = 25, threshold = 1.6. The blue triangles are above the threshold.

Identifying the Tail Regions with LOF

A straightforward approach to separate the tail regions from the flat region would be to set two thresholds, one for the upper limit and one for the lower limit (we denote this method THRES). However, as we show in Section 4.5, these thresholds can be difficult to set as the variability in the traffic volume between different days and locations would have an impact on the *Growth Rate* values, and setting fixed thresholds would mis-identify the tails. To this end, we observe that the tail regions have much lower data density than the flat region, and although *Growth Rate* values may change between days, the density boundaries between dense and sparse data may be a better indicator of the tail and flat regions. Therefore, we aimed to find a method that can exploit the differences in the density of the data, and in this study we use the Local Outlier Factors (LOF) [22] algorithm to select Emerging n-Edgesets. As we show later in Section 4.5, the LOF method is more robust in identifying the correct boundary by comparing the data density of each region, and consequently it out-performs the THRES method. For completeness, we briefly introduce the LOF algorithm.

The LOF algorithm was initially used to identify outliers in datasets [22]. (For an explanation of outlier detection, please refer to Section 4.2.1.) The LOF algorithm uses a density-based approach to identify outliers:

1. For each data point p, the distance between itself and its k^{th} nearest neighbour is calculated. We denote this value $k\text{-}dist(p)$ where k is a neighbourhood size parameter given by the user. A larger neighbourhood is less sensitive to noise but also increases the likelihood of misclassifying an outlier as normal data. Therefore, k may require some tuning to suit the dataset under consideration.

2. We compute the *reachability distance* between all pairs of points (p,q):

$$reach\text{-}dist(p,q) = max\{k\text{-}dist(p), dist(p,q)\} \tag{4.4}$$

where *dist(p,q)* is the Euclidean distance between p and q.

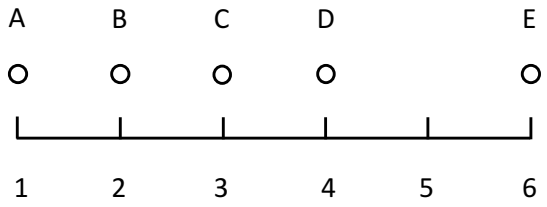


Figure 4.4: An example to illustrate the LOF algorithm

-	A	B	C	D	E
A	-	2	2	3	5
B	2	-	2	2	4
C	2	2	-	2	3
D	3	2	2	-	2
E	5	4	3	2	-

Table 4.2: The reachability distance between all pairs of points in Figure 4.4

3. We compute the *local reachability density(lrd)* for a point p:

$$lrd(p) = \left(\frac{1}{k} \sum_{q \in K} reach\text{-}dist(p,q) \right)^{-1} \tag{4.5}$$

where the sum is taken over the set of all k nearest neighbours of p, K

4. The LOF score for a point p is

$$LOF(p) = \frac{1}{k} \sum_{q \in K} \frac{lrd(q)}{lrd(p)} \tag{4.6}$$

5. Points with LOF scores higher than a user-defined threshold V will be selected as outliers.

LOF - Example

We illustrate the use of the LOF algorithm with a simple 1-dimensional example in Figure 4.4. Points A, B, C and D are 1 unit distance apart whereas point E is 2 unit distance from point D. We calculate the LOF scores step by step as follows:

1. We set the neighbourhood size k to 2, and so the KNN distances of the five points are $(2,2,2,2,3)$.

2. The reachability distance for pairs (A,B) is $max(2,1) = 2$, and for pair (A,E) is $max(2,5) = 5$. Similarly, we compute the reachability distance between all pairs (table 4.2).

3. The local reachability density for point A is thus: $\left(\frac{1}{2}(2+2)\right)^{-1} = 0.5$, and similarly for other points: $lrd(B) = 0.5$, $lrd(C) = 0.5$, $lrd(D) = 0.5$, $lrd(E) = \left(\frac{1}{2}(2+3)\right)^{-1} = 0.4$.

4. The LOF scores for points (A,B,C,D,E) are $(1,1,1,1,1.25)$.

5. Therefore, setting a threshold at 1.1 will be able to separate point E with the rest of the points.

Using LOF in Detecting Emerging n-Edgesets

The procedure of using LOF in detecting emerging n-Edgesets is as follows:

1. Find the LOF scores for each n-Edgeset in all n-Edgesets based on the sorted *Growth Rate*

2. Find all n-Edgesets with LOF scores larger than a threshold *thres*. These are the Emerging n-Edgesets (*EnES*).

3. If an *EnES* has a *Growth Rate* > 1, add it to the list of *EnES* with increased traffic. If its *Growth Rate* < 1, add it to the list of *EnES* with decreased traffic.

Although the LOF method is used, this is not anomaly detection. In anomaly detection, there is no prior knowledge about the time of occurrence of the anomaly, whereas we compare the traffic system before and after a traffic intervention, such as a road closure. Recall that our aim is to evaluate the effect of the traffic intervention, rather than detecting its presence. Figure 4.3(b) shows the resulting LOF scores for the 2-Edgesets in Figure 4.3(a), where the tails with low density have been successfully discovered and are marked with blue triangles. LOF requires two parameters, k, the number of neighbours, and *thres*, a threshold. In Section 4.5 we show that our method is robust against variations in these two parameters.

4.4.4 Mining Frequent Emerging Networks

The Emerging n-Edgesets reveal the paths that have been affected before and after a traffic intervention. As shown in Figure 4.3(b), many Edgesets can be affected. To identify the region

Algorithm 4.1: MineFreqNetwork

Data: $EmSet$: Emerging n-Edgesets, $RNGraph$: Road Network Graph, $Thres$: frequency threshold
Result: $FENet$: Frequent Emerging Network

```
1  begin
2      Uniq ← unique edges in EmSet
3      FENet ← empty list
4      forall the edge ∈ Uniq do
5          // count frequency of each edge
6          c ← frequency of occurance of edge ∈ EmSet
7          if c > Thres then
8              FENet.add(edge)
9      forall the edge ∈ FENet do
10         // find neighbours that are also in Network
11         Neighb ← neighbours of edge ∈ RNGraph
12         if Neighb ∩ FENet == ∅ then
13             FENet.remove(edge)
14     return FENet
```

that has suffered the most significant impact, we proposed a method for finding the Frequent Emerging Network in the Emerging n-Edgesets (Algorithm 4.1). We require that an edge in the list of Emerging n-Edgesets is a part of the Frequent Emerging Network if and only if it has occurred at least c times in the Emerging n-Edgesets and has a neighbouring edge that is also frequent. The advantage of this method is that it can extract the core part of the network that is most affected. In a real trajectory dataset (more details in Section 4.5.1), noise in the trajectories can cause issues in defining the boundary between affected and unaffected areas. By applying frequent itemset mining, this problem can be mitigated since the same noise is unlikely to occur repeatedly. In Algorithm 4.1, the *FEN* is initialized as an empty list (line 3). From lines 4 to 8, the frequency of occurrence of each edge is counted in the Emerging n-Edgeset, and the edges that occur more than c times are added to the *FEN*. From lines 9 to 13, edges that do not have a single neighbour that is also in the *FEN* are pruned out.

4.5 Experimental Evaluation

In this section, we present two case studies to evaluate the effectiveness and robustness of our algorithms. We first present a real-life case study using taxi GPS trajectories, as well as two traffic simulations that have been generated using a microscopic traffic simulator.

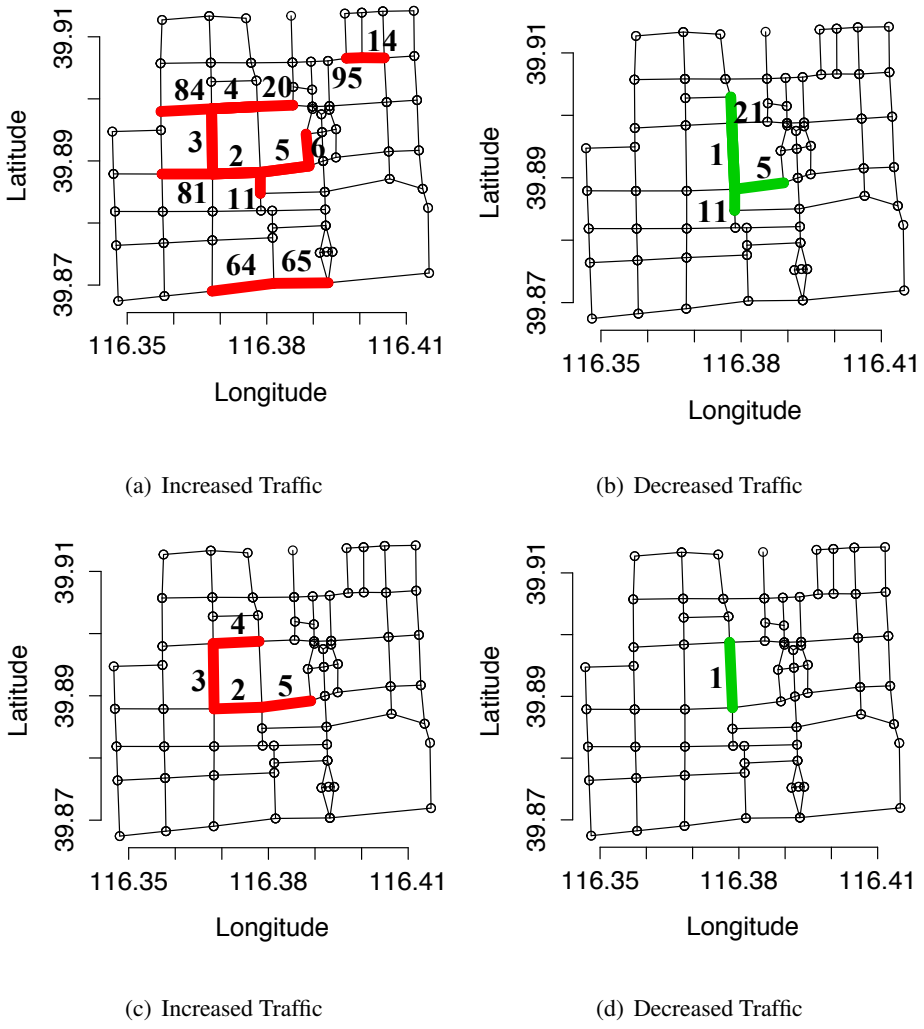
4.5.1 Real-life Case Study

We used a Beijing taxi GPS trajectory dataset, which is publicly available from Microsoft Research Asia [148][147]. The dataset consists of trajectories generated by 10,357 taxis travelling in Beijing and its surroundings over a seven-day period (02/Feb/2008 to 08/Feb/2008). The road map of Beijing is obtained from openstreepmap.org [106]. During the seven days, one road closure event is examined in this work.

Road closure event: South Xinhua Street was closed daily between 8:00 and 18:00, from 07/Feb/2008 to 11/Feb/2008 due to Chinese New Year. Therefore, two days of the road closure event are captured in the taxi trajectory dataset.

We average the data of the closed days (07/Feb to 08/Feb) and mine emerging patterns from the average of the two previous days without the road closure (05/Feb to 06/Feb). A smoothing step is required since taxi trajectories can suffer from noise issues caused by insufficient data on smaller roads due to the GPS sampling rate. To further reduce the effect of noise, we filter and remove the Edgesets with fewer than 10 cars in total during the two days. Since our focus is on the major roads, the above steps will not affect our results.

Although the Beijing dataset has been used in various past studies [161–164], the data mining problem being studied in this work is different from the literature, and therefore a direct comparison of our mining approach with the results in the literature is not possible. Furthermore, as the data were captured in the past, it is not possible to obtain ground truth observations of the traffic flow information without records from the traffic authorities, which are not available. Therefore, in Section 4.5.2, we evaluate our approach in a controlled simulation environment and compare the proposed method with the THRES baseline (described in Section 4.4.3).



(a) Increased Traffic

(b) Decreased Traffic

(c) Increased Traffic

(d) Decreased Traffic

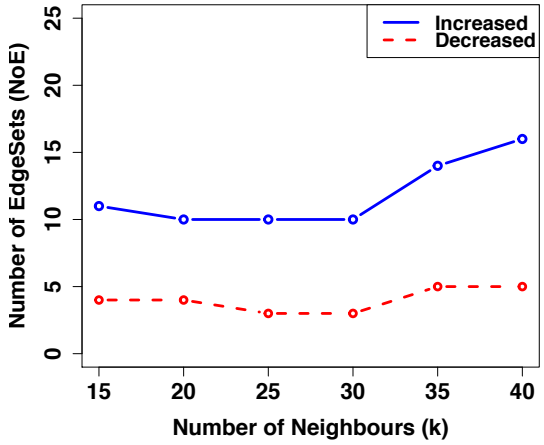
Figure 4.5: Emerging 2-Edgesets (E2ES) and Frequent Emerging Network (*FEN*) extracted using these Edgesets. (a) E2ES with increased traffic. (b) E2ES with decreased traffic. (c) *FEN* with increased traffic. (d) *FEN* with decreased traffic.

Effectiveness

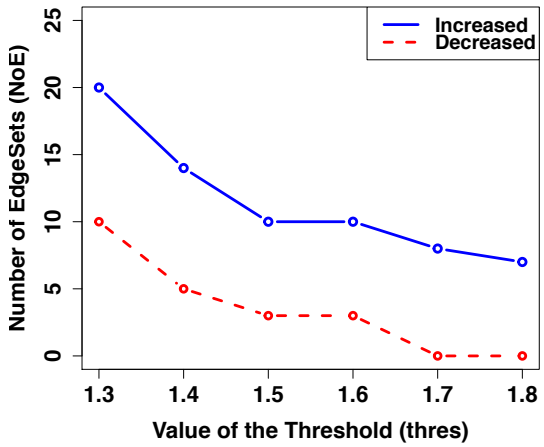
Figure 4.5 shows the emerging patterns that we found due to the road closure. Figure 4.5(a) depicts the 2-Edgesets with increased traffic after the road closure (red), while Figure 4.5(b) illustrates the decreased traffic (green). Edge 1, shown in green in Figure 4.5(b), was the road that was closed. As expected, the area surrounding the closed road has increased traffic levels since cars would have had to detour around the closed road. The paths that traverse the closed road, for example, path 5 to 1 and 1 to 21, have reduced traffic after the road closure, which is also expected as no cars can travel through the closed road. Figure 4.5(c) and 4.5(d) depicts the Frequent Emerging Network (*FEN*) extracted from the Emerging 2-Edgesets in Figure 4.5(a) and 4.5(b). For roads with increased traffic (Figure 4.5(c)), the edges adjacent to the road closure are part of the Frequent Emerging Network. This is expected since the impact should be inversely correlated to distance. For decreased traffic, only the closed road is reported. Therefore, the overall impact of the road closure on the traffic is mostly localized. Some isolated Emerging Edgesets in Figure 4.5(a) can be observed, such as $\{64, 65\}$ and $\{95,14\}$. Due to the unsupervised nature of the problem, whether these Edgesets are true positives or noise is unknown. However, as they are infrequent and removed from the *FEN* (Figure 4.5(c)), they are unlikely to have a large impact on the traffic. Therefore, these Emerging Edgesets are likely to be noise, and the robustness of *FEN* is demonstrated.

Parameter Sensitivity

For the LOF algorithm, two parameters are essential, k, the number of neighbours; and *thres*, the threshold of LOF scores. We varied one parameter while fixing the other, and compare the number of Edgesets (NoE) selected (Figure 4.6). To choose k, one typically starts from a small positive integer and then increases k. This is because k defines the size of the neighbourhood that a data point can compare against, and a larger k provides a greater smoothing effect. Starting from a small value will reduce the likelihood of over-smoothing, which will result in a greater extent of information loss. In Figure 4.6(a), NoE only varies slightly when k is between 15 and 30. Therefore, our results are not sensitive to the selection of k in that range, and k is set to 25 for future experiments. To choose *thres*, one starts from a value slightly larger than 1.0 and then increase



(a) Growth Rate



(b) LOF Scores

Figure 4.6: Number of 2-Edgesets selected by LOF with varying parameters. (a) Fixing *thres* = 1.6 and vary k. (b) Fixing $k = 25$ and vary *thres*.

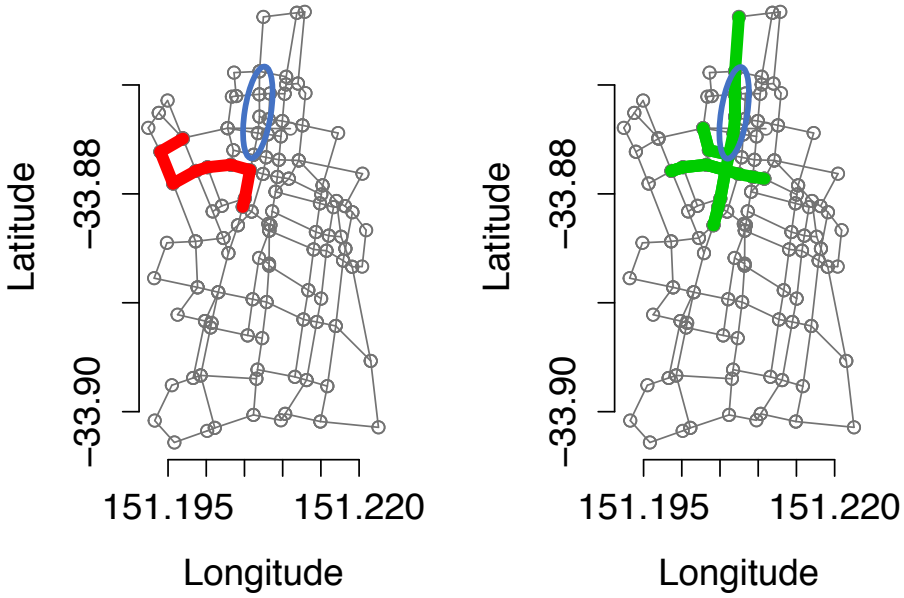
Figure 4.7: A screenshot of the traffic simulator

thres. This is because the majority of the Edgesets usually have LOF scores of near 1.0, which indicates that their *Growth Rate* values are similar. When *thres* increases from 1.0, NoE decreases steadily. Finally, NoE tends to converge when the difference between adjacent LOF values is large, and *thres* should be set in this range. Therefore, from Figure 4.6(b), *thres* is selected to be 1.6, and this value for *thres* is used in our later experiments.

During our experiment, we could easily extract emerging patterns for 2 and 3-Edgesets. However, due to the low sampling rate of GPS devices (1 to 10 minutes), it is difficult to extract meaningful patterns for Edgesets longer than three edges. In the next section, we show the results of using a traffic simulator, which overcomes this limitation.

4.5.2 Traffic Simulation

We have used a microscopic traffic simulator to validate our algorithms [77]. The simulator can perform large-scale and highly detailed traffic simulations (Figure 4.7). Vehicles are individually modelled to simulate realistic car-following and lane-changing behaviours. Various traffic rules

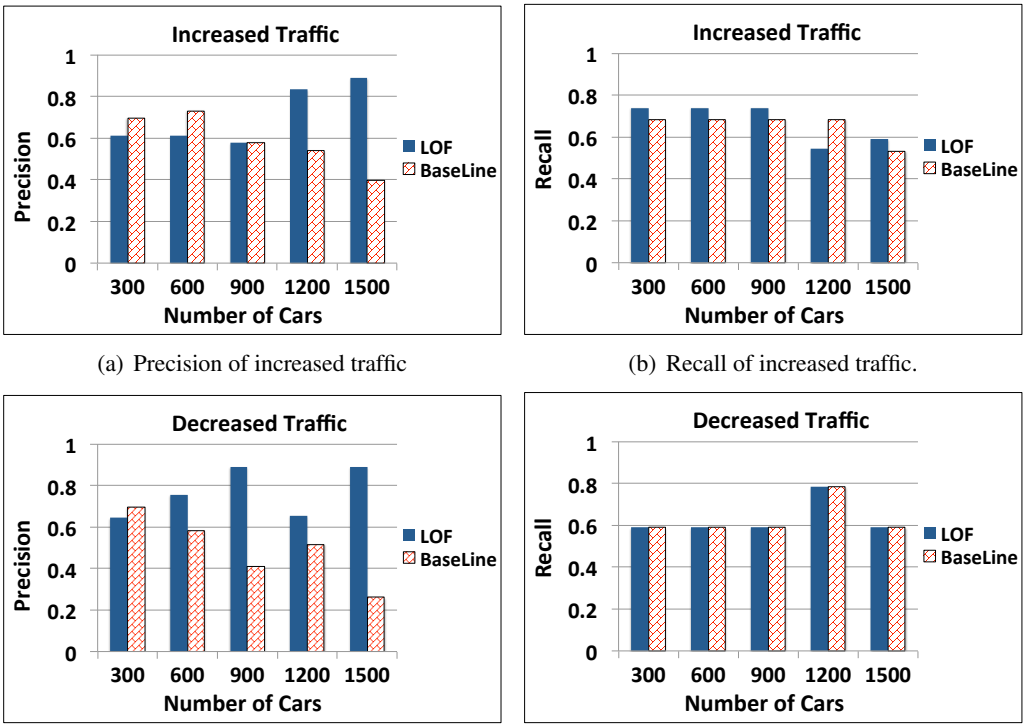


(a) Increased Traffic (b) Decreased Traffic

Figure 4.8: Simulation results and Frequent Emerging Network for 4-Edgesets in Sydney with George Street closed. (a) Increased Traffic. (b) Decreased Traffic.

are implemented, such as how to give way to trams at tram stops. The simulator can also simulate traffic lights, whose timing can be controlled by static or dynamic strategies. In each experiment, the road network was extracted from openstreetmap.org. The traffic is injected into the simulated area from three points on each edge of the map (totalling eight injection points), and vehicles are instructed to drive to the opposite edge through the simulated area. For example, a vehicle generated from the top edge will drive to the bottom edge. Dijkstra's algorithm [42] was used as the routing algorithm for the cars. In each experimental session, the total number of vehicles remaining in the area must be specified, which determines the congestion level of the roads. The more congested the roads are, the slower the vehicle speed and the longer the simulation must be run. We extracted two road networks from Beijing and Sydney with a similar area, and we used 300 to 1500 cars with an increment of 300 cars to simulate light to heavy traffic throughout the region. The speed of each vehicle is coloured by green, yellow and red representing 60-40km/h, 40-20km/h and 20-0km/h respectively. The number of cars was determined by observing the percentage of cars in each colour. With 300 cars, most cars are green and when there are 1500 cars, the vast majority are red. We simulated 1300 steps in each experiment since the traffic reaches a steady state for all levels of congestion at this value. All other settings were left at the default values given by the simulator [77], such as the minimum distance between two vehicles, timing of traffic lights, speed limit and lane changing rules.

We simulated road traffic in the City of Sydney, where the road topology is much more sophisticated than Beijing. We closed a few segments of George Street, which is one of the main streets in the CBD of Sydney. Since the sampling frequency of data can be set to a high value and no longer poses a limitation, we easily extracted Emerging 2,3,4-edgesets. The results of extracting the Frequent Emerging Network of 4-Edgesets for Sydney are shown in Figure 4.8. The experiment was initially run with 900 cars to simulate a medium level of traffic. The closed segments of George Street are circled in blue. Figure 4.8(a) shows that the edges near the closed street have increased traffic levels (red), while there is a decrease in the traffic on the closed road (Figure 4.8). The simulation results are consistent with our real-life case study using taxi GPS trajectories from Beijing.



(a) Precision of increased traffic

(b) Recall of increased traffic.

(c) Precision of decreased traffic.

(d) Recall of decreased traffic.

Figure 4.9: Precision and recall of LOF and the baseline method under different settings of traffic load in the simulation. The baseline is the THRES method mentioned in Section 4.4.3
. (a) Precision of increased traffic. (b) Recall of increased traffic. (c) Precision of decreased traffic. (d) Recall of decreased traffic.

Robustness

We evaluate the robustness of the LOF-based method using different numbers of cars (300, 600, 900, 1200, 1500) in the simulation. The common Edgesets are found and treated as ground truth. Using the common Edgesets, we evaluate the precision and recall of each single experiment for the case of increased traffic (Figure 4.9(a) and 4.9(b)) and decreased traffic (Figure 4.9(c) and 4.9(d)). The LOF-based method was compared with the THRES baseline, which sets an upper and a lower threshold to the *Growth Rate* to select Emerging Edgesets (Section 4.4.3). For precision, it can be seen that overall, the LOF-method outperforms the baseline. When traffic volume is high (1500 cars), the precision of the LOF method is 0.89 for both increased traffic and decreased traffic, whereas the baseline method has a precision of only 0.39 for increased traffic and 0.26 for decreased traffic. This implies that the variance in the value of *Growth Rate* can be large for a large number of cars and the results of the baseline method can be sensitive to the threshold chosen. However, the LOF method is not affected since the variance in the data density of *Growth Rate* can be much smaller than the variance in the value of *Growth Rate*. When the traffic volume is very low (300 cars), the precision of the LOF method (0.61 for increased traffic, 0.64 for decreased traffic) is slightly below the baseline (0.69 for both increased and decreased traffic). This might be caused by the fact that a small number of cars are less likely to traverse all the edges in the whole road network. Consequently, traffic volume calculations can become noisy, and the performance of the LOF method can be slightly affected. For other experiments, the precision of the LOF method is at least comparable (600 cars) or better than the baseline (900 and 1200 cars).

Figure 4.9 shows the recall of each experiment for the case of increased traffic (Figure 4.9(b)) and decreased traffic (Figure 4.9(d)). The LOF-based method and the baseline show comparable results for both increased and decreased traffic. This is expected since the Edgesets being significantly affected by the road closure are usually adjacent to the closed road. Therefore, these Edgesets appear to be significantly more dominant than other Edgesets, and both methods are able to find them. The recall for decreased traffic for both methods is the same, which is also expected since the traffic reduction on the closed road is obvious (*Growth Rate* dropping to 0). From the evaluation of both precision and recall, it can be seen that the LOF-based method is more robust than the baseline.

4.5.3 Computational Complexity

The most computationally intensive part of our framework is the traffic volume calculation for each n-Edgeset. Let s denote the total number of n-Edgesets, and d denote total number of trajectory points in our database. Note that the number of n-Edgesets is constrained by the connectivity between edges in the road network. This problem is equivalent to finding multiple matches of s strings in d, which is a well-studied problem, and an average case of $O(s+d)$ time can be achieved using a hash table. For other parts of our framework, mining both Emerging n-Edgeset and *FEN* take $O(s)$ time and the overall time complexity is $O(s+d)$.

4.6 Conclusions and Future Work

We have proposed a Contrast Mining framework to assess the impact of traffic events on the road traffic system based on the trajectory data of vehicles. GPS trajectories were mapped to road segments, and were converted to time-stamped sequences. The sequences that have a significantly different frequency of occurrence before and after the event were extracted by computing the *Growth Rate* and Local Outlier Factor (LOF) score. We also used a graph-based approach to propose the algorithm MineFreqNetwork to identify the sub-network of a road system that is significantly affected by the traffic event. Our experiments using real-life taxi GPS data and simulated vehicle trajectories show that our approach outperforms a baseline algorithm in terms of robustness.

There can be a number of directions for future work. In this work, we manually extracted road segments and intersections from road networks to aggregate trajectories into sequences represented by their nearest edge number. This is possible with a small road network like ours (4km by 4km), although manual extraction of road segments may be unrealistic for a larger system. One possibility is to divide the map into grids and assign trajectories to grid cells, which is a widely-used preprocessing step (e.g., [143]). However, this approach comes at a cost of granularity and information can be easily missed when the grid size is not chosen properly. Alternatively, vector map data from openstreetmap already contain points and lines, which can be directly used to assign trajectories onto road segments. However, the curvature of roads means that a curved road will be split into many short linear segments, which can be too dense for a sparse trajectory dataset

sampled at a frequency of every one to three minutes. With this approach, many road segments will have zero trajectory points assigned to them, and no useful information can be extracted. Finding a better method to define appropriate nodes and edges will be an important problem in future work.

Another limitation of this work is that we cannot identify multi-level traffic such as underground tunnels and elevated highways. These roads require the GPS sensor to record elevation level as well as the latitude and longitude, which is currently not available in most GPS devices. One way to mitigate this issue is to use the direction of travel and previous GPS points, then infer the road by finding the best matching segment.

A third direction is to suggest alternative driving routes. In this work, even though we have identified increased and decreased traffic flows, we do not suggest an alternative route due to the scarcity of events and limited amount of GPS data. If real-time data are available, it will be interesting to combine real-time anomaly detection and event characterization so that a driver can be better informed to avoid traffic events.

A fourth direction is to explore the sensitivity of our approach to more subtle or distributed special events. So far in our evaluation, we have considered focussed events, when a single road segment has been completely closed. In practice, many events may cause a partial closure of traffic lanes over multiple road segments, such as during road works. In principle, our method is readily applicable to such scenarios. However, representative real-life GPS traces from such events will be needed to accurately assess the sensitivity of our proposed contrast mining framework to this type of scenario.

Chapter 5

Using Trajectory Features for Upper Limb Action Recognition

There is growing interest in using low-cost wearable sensors to model limb movement in applications such as stroke rehabilitation and physiotherapy. This chapter presents an algorithm for the detection and classification of arm motion in time series collected by wearable inertial sensors. Arm trajectory features are obtained from raw sensor data using a sensor orientation tracking algorithm and an arm model. The features are then used in a clustering-based classifier. In the classifier training stage, features are clustered using the K-means algorithm, and a histogram of key poses is generated from the clustering as a template for each class. In the recognition stage, new data are segmented and matched to the templates. Experiments on human subjects show that by using trajectory features in the proposed approach, we can achieve higher accuracy than a range of benchmark non-temporal classifiers. The publication arising from the work in this chapter is paper P3.

5.1 Introduction

USING wearable sensors in medical applications such as stroke rehabilitation and physiotherapy to model limb movement has attracted growing attention [111]. The increasing health care cost has motivated the development of remote health monitoring, and technologies such as the Internet of Things and sensor networks are playing increasingly important roles [95]. In stroke rehabilitation, for example, a patient usually undergoes two stages, hospital and home rehabilitation. After the onset of stroke, a patient first performs rehabilitation in a hospital where the recovery progress is closely monitored by physicians and feedback is immediately available to the patient. The patient is usually discharged after regaining basic movement capabilities such as walking, and rehabilitation usually continues at home to fully regain motor functions. Home

rehabilitation is usually unmonitored and the progress of recovery is difficult to assess. Typically, a patient routinely re-visits the hospital to seek feedback on the rehabilitation progress, which is costly and inefficient [110]. The use of wearable sensors to capture limb movement information and provide feedback to patients can be a cost-efficient alternative to frequent hospital revisits. For example, Zhang et al. have developed a home-based rehabilitation system that utilises cameras and motion sensors to capture patient movement [158]. Using this system, a patient can perform a set of prescribed exercises in front of a computer and the sensor data will be sent to the hospitals for the physicians to review and provide feedback on the rehabilitation progress. Such systems have saved the cost of travelling to and from the hospital. However, the range of actions that are performed can be different from daily activities. Moreover, the need to sit in front of a camera to perform certain exercises may lead to the effect of social facilitation [153], i.e., the person may act differently when being watched than when alone. The patient may feel the need to "work hard" during the session and the captured data can be distorted so as to not reflect the actual arm usage during daily life. To reduce this effect, it is desirable to capture movement data with wearable sensors during daily routines, rather than at a specific location, at a particular time and in front of a camera. As wearable sensors can be attached to the body over a long time-frame with ease, the activity profile can be captured with reduced user awareness. For example, smart phones and smart watches contain accelerometers and gyroscopes, which can be used to record the motion data of the user. The data can be transmitted via the IoT infrastructure to the hospital for further analysis and diagnosis.

In this context, one important problem of interest is to recognize specific actions performed by the person in daily life so that a physician can easily observe the patient's movement from the continuous data stream sent by the sensors. This can be modelled as a classification problem with stream data. The input is motion data recorded by wearable sensors, and the output is the type of actions completed by the user. Recognising the actions can allow a physician to visualise the activity levels of different body parts, and give quantitative feedback of the rehabilitation progress of the user. However, this is a difficult problem since actions are performed in an uncontrolled environment, i.e., no instructions are given to the person on when and how to complete daily tasks. This leads to the fact that the same action can be performed in a number of ways. For example, "eating" can be performed with the palm up or palm down depending on the individual

and the food. Although the arm may follow similar trajectories, the data collected on each axis of the sensors can vary significantly due to different orientation of the sensors, and hence the recognition accuracy can be unsatisfactory.

In this chapter, we focus on this problem of recognising human actions with wearable sensors. We propose the use of a 3D trajectory reconstruction algorithm to process the raw sensor data and extract trajectory features that can improve action recognition accuracy. We also propose a clustering-based classifier for use with the trajectory features, and compare the recognition accuracy with a range of benchmark classifiers.

The rest of the chapter is organized as follows. Section 5.2 reviews related work and defines the problem. Section 5.3 gives an overview of our action recognition approach, the hardware set-up and data collection procedure. Sections 5.4 to 5.6 describe our method in detail. In Section 5.7 we evaluate our methods and present future research directions.

5.2 Related Work

Recognising human actions with wearable sensors has received considerable attention from a number of research communities. Such systems consist of two key components: the use of sensors and the recognition algorithm. Depending on the specific application and the types of actions to be recognised, different sensor set-ups have been proposed to obtain features from various body parts. We summarise the key features of a number of representative papers in Table 5.1 and a thorough survey of the relevant literature can be found in [10, 81]. We review the literature from three perspectives in this section: (1) action recognition tasks, (2) feature extraction, and (3) classifier design.

5.2.1 Action Recognition Tasks

Different types of action recognition tasks have been studied in the literature, which requires various sensing devices, sensor counts, sensor placement set-ups and data collection methods. Most relevant studies (see Table 5.1) use only wearable sensors in action recognition while some authors have combined wearable sensors with microphones [94] and object sensors such as RFID tags [28, 94]. The wearable sensor can be the accelerometer unit embedded in a smart phone or a

Table 5.1: A summary of previous work

Ref	Recognition Accuracy	Activities Recognized	Classifier	No. Subjects	Sensor Positioning	Sensor Types
[116]	55% - 99%	Whole body	Non-temporal	2	waist	single 3-axial accelerometer
[14]	35% - 84%	Whole body	Non-temporal	20	arm, thigh, waist	2-axial accelerometers
[28]	53% - 86%	Whole body	Non-temporal	4	arms, feet, back	9-axial inertial units, object sensors
[127]	91%	Whole body	Non-temporal	7	thigh	single 3-axial smartphone accelerometer
[80]	55% - 95%	Whole body	Non-temporal	29	thigh	single 3-axial smartphone accelerometer
[94]	44% - 100%	Upper limb	Temporal	1	arms, waist	3-axial accelerometers microphone, object sensors
[120]	84% - 89%	Upper limb	Temporal	2	wrist	single 3-axial accelerometer
[73]	78%	Upper limb	Temporal	4	arms, back	9-axial inertial units
[41]	49% - 91%	Whole body	Non-Temporal	10	not standardised	single 3-axial smartphone accelerometer
[19]	75% - 90%	Whole body	Temporal	2	shoulder, wrist, waist, foot	3-axial accelerometers

dedicated inertial measurement unit (IMU) with an accelerometer, gyroscope and magnetometer. For accelerometer-only set-ups, the output is a stream of two or three dimensional vectors of acceleration values while for IMUs, the output is typically a stream of 9 dimensional vectors that consist of acceleration, rate of turn and sensor direction relative to the earth's magnetic field. As the number of sensors used and their placement can have an impact on the type of actions that can be recognised, we classify the action recognition tasks based on two types of sensor set-ups: (1) Single sensor and (2) Multiple sensor set-ups.

Single Sensor

One of the earliest action recognition approaches was proposed by Ravi et al. [116], who attached a custom-designed accelerometer board with a single tri-axial accelerometer to the waist of the subjects to recognise various activities. The authors captured data for eight activities from two subjects. Common daily activities such as walking, running, vacuuming, and brushing the teeth were recognised in their work. However, due to the limitation of using only one sensor, classification of fine-grained activities, especially those involving the upper limbs, were not studied. The activities were labelled in a semi-automatic fashion. The experimenter recorded the start time and used a stop watch to measure the duration of each action. The stop time was identified by adding the start time and duration. The data within 10 seconds of the start and stop time was discarded to reduce mis-labelling.

Rather than using a dedicated accelerometer unit as in Ravi et al. [116], advances in mobile computing have allowed researchers to use accelerometers on a smart phone to recognise actions. Kwapisz et al. [80] strapped an Android phone to the thigh of users and recorded data for six activities and body postures: walking, jogging, upstairs, downstairs, sitting and standing. Compared to the range of activities recognised by Ravi et al. [116], Kwapisz et al. [80] were unable to recognise upper-body actions, such as brushing the teeth, since the sensor was placed on the thigh and upper body features were difficult to obtain. Sun et al. [127] used a single smart phone to recognise seven activities: standing stationary, walking, running, bicycling, stepping upstairs, stepping downstairs and driving. Although the activities are similar to the ones recognised by previous authors, Sun et al. varied the location and orientation of the smart phone to simulate different scenarios that may occur in real life. For instance, a user may put the phone in the pocket on the hip or on the thigh,

and the phone may be placed with different orientations. The results obtained suggest that for a single-sensor set-up, the location of the sensor has a significant impact on the activities that can be recognised. A similar smart-phone-based framework was proposed in [7] where the authors focused on reducing the computational cost of the recognition algorithm with fixed-point arithmetic. Dernbach et al. [41] further considered the hierarchy of activities and categorised activities into simple and complex ones. Simple activities consist of one repeated action, for example, walking and running, whereas complex activities consist of a series of actions (for example, cooking is a complex activity that involves multiple actions). The placement of the phone was not standardised and was left to the convenience of the subject. A total of 15 activities were recognised in their work, which is significantly more than previous authors. Unfortunately, recognition accuracy for complex activities is much lower than simple activities, which is expected since only one smart phone was used and complex actions may require more information to be identified correctly.

In contrast to the above work, Sen et al. [120] used a single commercial wireless health monitoring device to recognise upper limb activities. The device is similar to the size of a wrist watch and was attached to the right hand of the subjects. Consequently, eating and drinking movements were recognised while lower limb and whole body movements were not considered.

As can be seen, single sensor set-ups using smart phones and watch-like devices can be convenient from a user's perspective and usually causes minimal amount of disturbance to daily life. However, due to the limited amount of data that can be captured, the range of actions being recognised can also be limited as sensor placement can play a major role in the data that can be obtained. If more complex activities are to be recognised, the accuracy is low due to insufficient information. Therefore, many authors have proposed systems that utilise multiple sensors, which are discussed below.

Multiple Sensors

Using multiple sensors can significantly improve the recognition accuracy of certain activities that involve multiple body parts. However, the inconvenience of wearing the sensors often restricts the applicability of such set-ups. Bao et al. [14] presented a framework that uses a semi-naturalistic approach to obtain user-annotated data to recognize daily activities. One bi-axial accelerometer unit was attached to the left arm, right wrist, waist, left thigh and right ankle of 20 subjects. The

authors designed a series of "obstacle courses" that were used to disguise the activities being captured and the subjects were asked to complete the obstacles without the supervision of researchers. The goal was to reduce subject awareness of monitoring so that the activities were performed in as natural a manner as possible. For example, to capture the activity "work on a computer", the obstacle was specified as "use the web to find out what the world's largest city in terms of population is". The subjects were asked to record the start and stop time of each activity to reduce the amount of labelling effort by the researchers. After completing the obstacle challenges, a more controlled experiment was performed where the subjects performed random sequences of 20 activities defined on a worksheet. Although a number of high level activities were recognised with very high accuracy (e.g., 97% for "working on a computer"), a total of five sensors were attached to the users, which can be somewhat inconvenient for daily use and long-term monitoring. For this reason, the authors further evaluated the effect of leaving only one or two sensors on the subjects, and the thigh and wrist sensors were found to be the most informative in differentiating the activities with the minimum reduction in recognition accuracy. This provides helpful information in terms of sensor placement. The above work mostly investigates the recognition of whole body actions such as standing, walking and vacuuming.

Similar to Bao et al. [14], Altun et al. [6] attached five sensors to various body parts and 19 activities were recognised, which is by far the largest activity set. The authors wrapped commercial IMUs on the arms, legs and the chest of the subjects and therefore, it is not surprising that a high classification accuracy was observed.

Lukowicz et al. [94] propose a framework to recognise workshop activities, including drilling, sawing and using a screwdriver, with five sensors and a microphone. Compared to Bao et al. [14], Lukowicz et al. [94] attach all sensors to the upper body of the subjects and the tasks have a large focus on arm and hand movement, which is more similar to our recognition problem. Although a microphone is helpful in action labelling, it is also more invasive than inertial sensors and may cause privacy concerns among the subjects. Therefore, this approach can be less practical for situations where extended monitoring is required. More recent work by the same group, Junker et al. [73], avoids the use of a microphone. Their work focused on the identification of actions in a sensor stream, which combines data segmentation with action recognition. However, to obtain accurate results, person-specific data were collected during the training stage for each subject,

which can be time-consuming and unscalable. Sen et al. [120] proposed the use of an arm model to capture features associated with the eating and drinking actions. However, only one sensor was attached to each wrist, which limited the range of actions that could be detected.

Most of the above studies conduct data collection in a laboratory environment. Bicocchi et al. [19] collected a real-life dataset where two users wore four sensors for one day while going about their normal daily routines. The ground-truth activities were recorded and annotated by a separate application. However, as only a high level description of the tasks is available, e.g., work at the computer and take something from the fridge, the detailed actions involving the body parts are unclear.

In this chapter, we only study upper limb action recognition for physiotherapy applications and hence lower body sensors are not required. Moreover, we require more information on the upper limb activities than whole body postures and therefore we place two sensors on a single arm, one on the wrist and one on the elbow. Combining with an arm model, this set-up allows the detailed arm trajectory features to be extracted and classification accuracy to be improved (Section 5.3).

5.2.2 Feature Extraction

Feature extraction is one of the most important steps in the design of accurate recognition systems. The feature extraction problem in action recognition can be stated as follows. Given a matrix of discrete sensor readings $X \subset \Re^{N \times M}$ where the row number N is the total number of samples and the columns represent raw sensor readings from M sensor channels, an action is a sequence of observations with M columns. The action is labelled by an action class C_i in a total of $\{C_1, C_2, ..., C_P\}$ action classes studied. Although the raw data matrix X can be used directly in classifying actions, it often contains noise, correlated columns or perhaps drifted means, which is sub-optimal to be used directly in a classifier. Feature extraction aims to find a transformed or reduced representation W from X for each action such that the columns of W, or the "features", are discriminative for each action class. This means the feature values for actions within the same class are similar and between different classes are dissimilar.

As the sensor data usually comes in as streams, feature extraction may be combined with data segmentation to transform the input data into a feature matrix. Early works in action recognition typically use a sliding window approach to segment actions [14, 116]. The name sliding window

originates from the fact that the segmentation appears as sliding a time window over the data stream. There are two windowing approaches, overlapping or non-overlapping. The captured input data stream is stored in a first-in-first-out buffer with length W, and when the buffer is filled, features are computed from the segmented stream. If raw data are completely discarded after feature extraction is finished, a non-overlapping approach is used. This segmentation scheme has the advantage of computational efficiency as the data are only processed once. However, the choice of windowing size is a non-trivial problem as a small window may only partially capture an action while a large size may combine two actions into a single window. Therefore, the recognition accuracy of non-overlapping windowing approaches can suffer greatly from a poor choice of window size. If a certain fraction of data are kept in the buffer to combine with newer incoming data, it is called the overlapping windowing approach. This method has been demonstrated to be more robust with respect to the choice of window size [14, 116]. However, as the features must be computed for each new window, the computational cost is higher than the non-overlapping approach. Therefore, computationally efficient features are usually used in overlapping approaches.

Bao et al. [14] extracted four features, mean, energy, frequency domain entropy and correlation. The mean feature is the average acceleration of the action over the window. The energy feature is the sum of the squared magnitude of the discrete Fast Fourier Transform (FFT) components of the acceleration signals, which captures the volatility of the action. Frequency-domain entropy [35] is a feature that computes the normalized information of the discrete FFT components of the raw sensor data. For example, an action with continuous motion, such as biking, will have few high frequency FFT components, and therefore a low entropy. Running, on the other hand, may result in many abrupt acceleration changes. The high frequency components of FFT will have large magnitudes and therefore higher entropy. Correlation is a feature that characterises the linear dependency of two sensors, which is computed as $corr(x,y) = \frac{cov(x,y)}{\sigma_x \sigma_y}$ where $cov(x,y)$ is the covariance and σ is the standard deviation. In addition, Ravi et al. [116] used the standard deviation of the acceleration signals $\sigma_x = \sqrt{\frac{1}{W}\sum_{i=1}^{N}(x_i - \mu)^2}$ as a feature along with mean, energy and correlation. A similar set of features have been used by Sun et al. [127] to recognise daily activities with a smart phone. Kwapisz et al. [80] further investigated four features, mean absolute difference, magnitude of acceleration, time between peaks and binned distribution.

The above features have been successfully used in the recognition of whole body daily activ-

ities such as walking, bicycling and running upstairs. However, for activities with finer details, especially for the upper limbs, these features can be ineffective. The reason is that individuals may perform upper limb activities with different accelerations and frequency domain profiles while achieving the same outcome. For this reason, research has been conducted on using arm models to extract positional features for upper limbs. Junker et al. [73] used pitch and roll angles of the arm to identify gestures. However, their work relied on collecting person-specific data for training, which is not scalable. Sen et al. [120] used an Extended Kalman Filter and an arm model to identify eating and drinking. However, only one sensor was attached to each wrist of the arms, and therefore only eating and drinking actions were studied in their paper.

5.2.3 Classifier Design

Two types of methods have been used in action recognition: temporal and non-temporal classifiers. Temporal approaches explore sequential information in the incoming data stream and make predictions based on the order of occurrence of signal levels. Non-temporal approaches discard the sequential information and assume that the features extracted from the segmented data (discussed in the previous section) have sufficient discriminative power such that the action classes can be separated by treating each data segment as a point in the feature space. We review the two classes of methods next.

Bao et al. [14] investigated the use of several well-known non-temporal classifiers including decision tables (DT), nearest neighbours (KNN), decision trees (DTr), and Naive Bayes (NB) in action recognition. The authors used the C4.5 algorithm [113] as the implementation of decision tree, which can be found in the Weka Machine Learning toolkit [62]. For their dataset, C4.5 performs the best with both user-specific (71%) and leave-one-subject-out (84%) evaluation results. Nearest neighbours slightly under-performs with a 2% accuracy for both cases while the other algorithms show significantly worse results. It is interesting to note that Naive Bayes performs badly in their experiment with accuracies of only 34% and 52%, losing around 30% to C4.5. This disagrees with the results by other authors. Ravi et al. [116] evaluated the above algorithms as well as support vector machines (SVM), and ensemble methods such as boosted trees and plurality voting. Naive Bayes is the second best-performing algorithm on average, with plurality voting being the most accurate classifier. As NB assumes that all features are independent, correlation

among features might have led to the vastly different results by the two groups, which is a core disadvantage of NB. Decision trees performs well on average for both datasets, although it is difficult to be used in online training. K nearest neighbours also performs well for both datasets, and KNN is an instance-based classification method which requires no training. The drawback of using KNN is in the recognition stage as a similarity score to every training sample must be computed and sorted, which is an $O(N^2)$ operation. If the training data is large, the recognition time can become unacceptable for real-time applications. To counter this problem, indexing methods such as the KD-tree have been proposed [100]. SVM is a powerful machine learning algorithm which is used frequently in the computer vision community in combination with deep neural networks to recognise objects such as hand-written digits [71, 105]. However, the computational cost to train an SVM is relatively high. If the training and recognition are implemented on board by a low-cost sensor node, SVM would be an inappropriate choice. The above non-temporal classifiers require both valid and invalid actions as training examples, and invalid actions can be difficult to model.

Temporal classifiers assume that correlation exists between sequential sensor readings and exploits such information to make decisions. The most commonly-used temporal classifier in action recognition is the Hidden Markov Model (HMM) [114]. An HMM makes the Markov process assumption on the system, i.e., the next state of the system is only dependent on the current state and not previous states. For a simple Markov model, the states are directly visible to the observer, whereas for HMMs the states are hidden and must be learned from the data (hence the name Hidden Markov Model). Therefore, HMMs usually require a relatively large amount of training data to learn the model, and the user may become annoyed when being asked to repeat the same action many times. Lukowicz et al. used HMMs to recognise workshop activities [94]. A set of eight activities such as hammering and sawing were recognised and one HMM was trained for each activity. Raw acceleration data were fed into the HMM training algorithm and the number of states were selected manually by the researcher for each activity. This training process can be inconvenient if more activities are to be recognised. Junker et al. [73] used pitch and roll features to spot and segment actions before using HMMs to classify the actions. To tackle the challenge that the classifier requires invalid actions to function properly, the authors instructed users to perform additional actions related but slightly different from the desired action. This increases the burden of data collection by the user.

To improve action recognition, we have identified two open challeges: 1) can the use of arm trajectory features improve the accuracy of action recognition, and 2) how to design a classifier that requires only a small number of valid training examples and is well-suited for the features.

5.2.4 Problem Statement

In this chapter, we are motivated by the challenge of monitoring the rehabilitation progress of stroke patients in their home environment, particularly for their upper limbs. We focus on the problem of action recognition using motion data collected from wearable sensors. Specifically, we aim to address these two challenges: (1) feature extraction from raw sensor data, and (2) the design of a classifier that can accurately classify the actions. We are particularly interested in taking full advantage of 3D arm motion reconstruction algorithms [128] to extract high level trajectory features. We define an action to be a simple gesture, which lasts in the order of a few seconds (for example, reaching for a cup on the table).

We make the following assumptions: (1) sensors are always worn at the same position and orientation; (2) the arm length for an individual is constant; (3) arm muscle contraction and relaxation is negligible compared to arm movement; (4) the magnetic field is uniform around the arm.

5.3 Overview of Our Approach

Our action recognition approach consists of four main stages: (1) trajectory feature extraction, (2) clustering-based training, (3) candidate segmentation, and (4) candidate recognition (see Figure 5.1).

Raw data are organized into a matrix $S \in \Re^{N \times M}$ where each row is a *measurement sample*, a single instance of measurement values captured by the sensors (giving N measurement samples in total), and each column is a data channel from the sensors (tri-axial accelerometer, gyroscope and magnetometer). Each *measurement sample* is labelled with its class $y \in \{$*null, action1, action2...*$\}$, where $\{$*action1, action2...*$\}$ denote valid actions (actions of interest) and $\{$*null*$\}$ denotes other invalid actions. The matrix S is then processed by a 3D arm trajectory reconstruction algorithm [128] to obtain trajectories and orientation of the arm, from which the trajectory feature

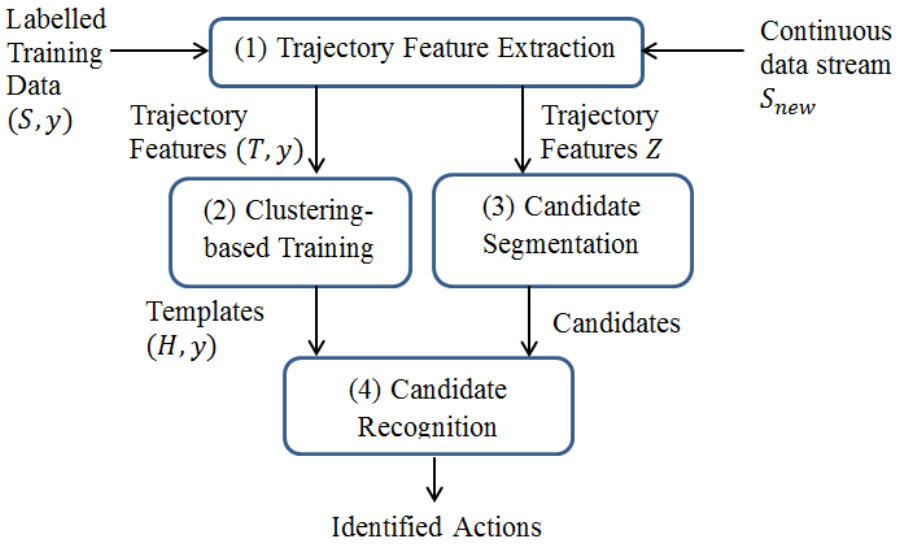


Figure 5.1: Overview of our action recognition framework.

matrix $T \in \Re^{N \times P}$ is computed (Section 5.4). The trajectory features are then used to train a k-means-clustering-based classifier, where the clusters represent the *key poses* of the actions (see Section 5.5). Histograms of the frequency distribution of all classes of actions in the clusters are constructed as a matrix $H \in \Re^{c \times (k_m+1)}$, where c is the number of actions and k_m is the number of clusters (see Section 5.5). These histograms represent the distribution of *key poses* for each class of actions and are used as the templates for classification. When recognizing new data, raw sensor data $S_{new} \in \Re^{W \times M}$ is also first converted to trajectory features $Z \in \Re^{W \times P}$. Candidate segmentation is performed to identify the existence of actions and to find the start and end of each action. The candidates are then classified using the histogram templates (see Section 5.6).

5.3.1 Data Collection

Data were collected from four healthy subjects (aged 23 to 40) using a commercial Xsens MT system at a 50-Hz sampling rate. Each sensor consists of a tri-axial accelerometer, gyroscope and magnetometer. Two sensors were attached to each subject, one on the wrist and one on the elbow (Figure 5.2). Each subject was given three tasks, {*eating, drinking, horizontal reaching*}, and asked to repeat each task five times with their right arm. Various food and dining utensils were used to increase the variability of motion (Table 5.2). All the subjects were right-handed and

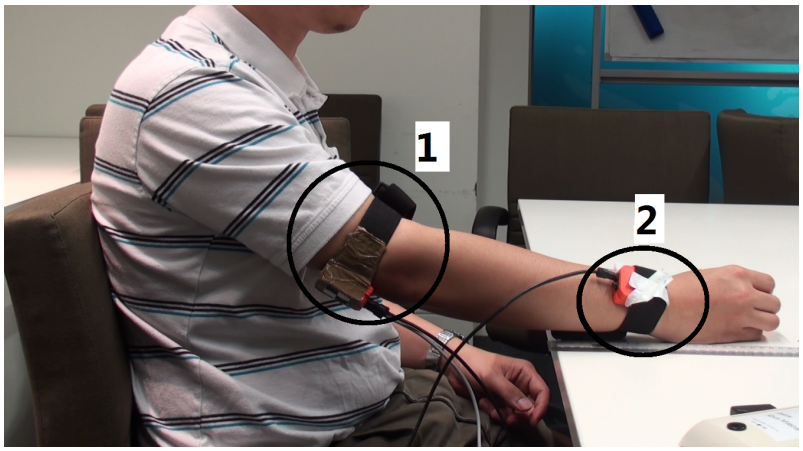


Figure 5.2: Experimental set-up with sensors attached to the wrist and elbow.

Table 5.2: Actions performed during data collection

Subject 1	**Subject 2**	**Subject 3**	**Subject 4**
Horizontal Reaching	Horizontal Reaching	Horizontal Reaching	Horizontal Reaching
Drinking	Drinking	Drinking	Drinking
Eat a cake with a spoon	Eat a hash brown with hand	Eat mandarin pulp with hand	Eat meat with chopsticks

asked to perform the actions as naturally as possible. The experiments were captured with a video camcorder to provide the ground truth and sensor data were labelled based on the videos. Three classes were labelled: {*null, eating and drinking, horizontal reaching*}. Eating and drinking were labelled as the same class due to their high similarity; invalid actions were labelled as {*null*}. Note that while the use of four subjects in this study is a somewhat small sample, it is similar or larger than the number of subjects used in 6 out of the 10 comparable studies reported in the literature in Table 5.1.

5.4 Trajectory Feature Extraction

Feature extraction is important in designing an accurate action recognition system. As the training and testing data are usually performed by different subjects, variations often occur in the way individuals perform the same action. For example, for the action *eating*, the data captured by

sensors on two subjects can exhibit vastly different features even though the conceptual definition of the class is unchanged. To obtain high recognition accuracy, features should be robust to variations between subjects but consistent within the same class. Although raw sensor signals can be used directly as features for classification, the recognition accuracy will be compromised as orientation differences in performing the actions will have significant effects on the features. Using trajectory features for action recognition can be superior to using raw sensor signals since orientation differences are removed and only the travel path of the arm is considered. The disadvantage of using only trajectory features is that actions associated with purely arm rotation cannot be identified such as turning a door knob. However, that can be mitigated by combining the raw data with trajectory features. In Section 5.7.3, we compare the recognition performance of trajectory features and raw sensor signals and show that an improvement is indeed observed.

We extract trajectory features from raw sensor data using a sensor fusion algorithm [128]. There are many sensor fusion algorithms in the literature and the most well-known method to extract sensor orientation is the Extended Kalman Filter (EKF) [118]. However, the error in EKF tends to accumulate and frequent re-initialization is required [128]. To this end, we used a sensor fusion algorithm proposed in [128] that is specifically designed to track 3D orientation and is more accurate and stable than EKF. The algorithm performs two steps: 1) obtain the orientation of individual sensors; 2) combine sensor orientation with an arm model to obtain trajectory features. It is outlined as follows:

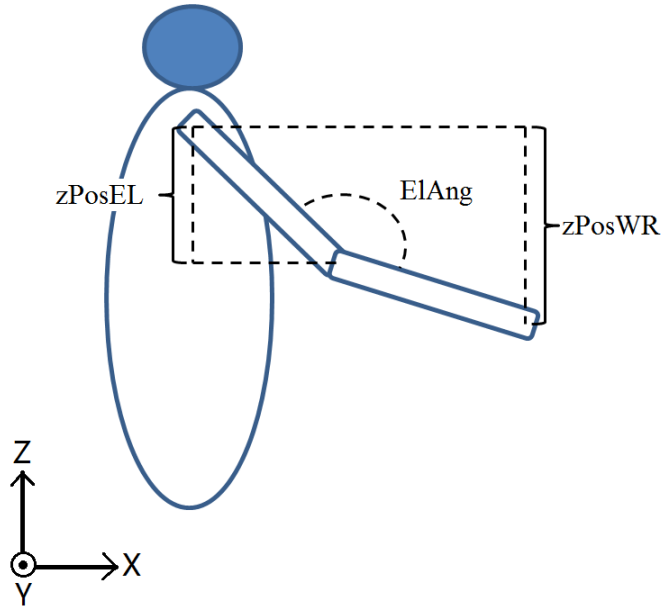
1. The objects to be tracked are two 3-by-3 rotation matrices, R_1 and R_2, which represent the orientation of the wrist and elbow sensors.

2. The matrix Von Mises-Fisher distribution [75] is used as a statistical model of the process noise.

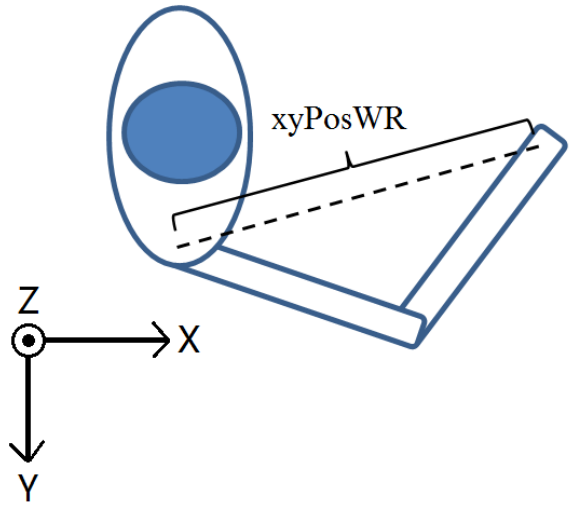
At each time step t:

1. The gyroscope measurements are modelled by a normal distribution and are used for computing prior distributions of R_1 and R_2 through Lie algebra.

2. Accelerometer and magnetometer data is then used to compute the posterior distribution of R_1 and R_2.



(a) Side view



(b) Top view

Figure 5.3: Illustration of the trajectory features.

Table 5.3: Trajectory features and notation

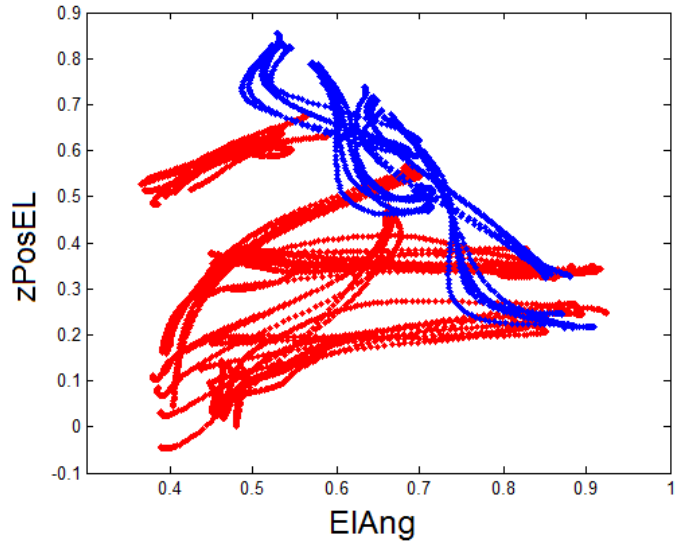
Feature Name	Notation
Elbow Flexing Angle	ElAng
Elbow Z Position relative to shoulder	zPosEL
Wrist Z Position relative to shoulder	zPosWR
Wrist Radial Position from shoulder	xyPosWR

A complete description is beyond the scope of this work and more details can be found in [128]. The orientation matrices are then used in an arm model similar to the work by Luinge et al. [93] from where we extract higher level trajectory features (Table 5.3 and Figure 5.3). The trajectory features are arranged in a matrix $T \in \Re^{N \times P}$ where each row is a sample taken at 50 Hz and each column is a feature.

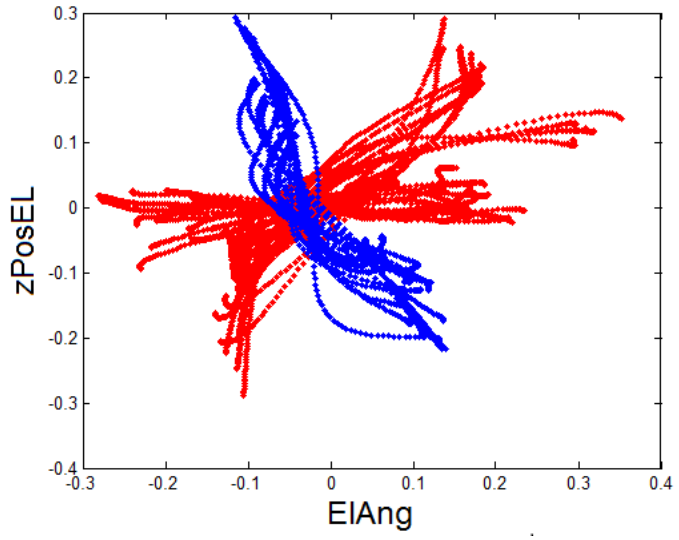
The feature matrix T can be used directly in classification. However, further processing is required to obtain higher accuracy. This is illustrated in Figure 5.4(a) where the projection of the trajectory feature space are shown for the set of {*eating and drinking, horizontal reaching*} actions performed by two subjects. It is desirable that the trajectories of the same class share more overlap than that of different classes since the overlap signifies similarity between trajectories. If the mean of each action is removed (Figure 5.4(b)), the distance between two actions of the same class decreases, and directional differences can also be observed between the trajectories of different classes. Hence, removing the mean of each action improves the discriminative power of trajectory features and thus is implemented in our algorithm. The improvement in classification accuracy of this mean-centring step is shown in Section 5.7.

5.5 Clustering-based Training

The aim of the training stage is to find distinct patterns for each class of actions using trajectory features. As discussed in Section 5.2, although HMM is promising, it requires a large amount of training data. Methods such as SVM require the "negative class", i.e., invalid actions except {*eating and drinking, horizontal reaching*} in our case, which are difficult to model. Therefore, we take a clustering-based approach where only a small number of valid training examples are required. The intuition is that by separating the training data into clusters, some clusters will rep-



(a) Original trajectories



(b) Mean-centred trajectories

Figure 5.4: Action trajectories before and after mean centring for two subjects. Red: eating and drinking actions. Blue: horizontal reaching actions.

resent *key poses* of each action, and if we count the frequency distribution of *measurement samples* in the clusters for each action, this distribution can be used to distinguish actions. This approach was first investigated by Wang et al. [139] to identify human motion in videos. Algorithm 5.1 shows the pseudo code of the training process.

After feature extraction, the trajectory feature matrix $T \in \Re^{N \times P}$ is obtained. All *measurement*

Algorithm 5.1: Clustering-based Template Construction

Data: $T \in \Re^{N \times P}$, $L \in \Re^{N \times 1}$, $L(i) = y \in \{null, action1, action2, ..., actionL\}$
Result: $H \in \Re^{Q \times (k_m+1)}$: Histograms of actions

```
begin
    Do k-means, k = k_m for all rows v in T if L(i) ≠ {null}
    for j ← length(y) − 1 do
        // each class
        for p ← 1 to k_m do
            // each cluster
            Count(p) ← the number of v in cluster p
        Build a histogram h ∈ ℜ^{1×k_m} using Count
        h ← h append 0
        H(j) ← h
    return H
```

Algorithm 5.2: K-means clustering algorithm

Data: X: N-by-P data matrix with N data points and P features, D: parameter to set the number of clusters, R: max number of iterations
Result: Y: cluster labels of length N, C: D-by-P matrix containing centroid locations

```
begin
    Randomly initialize Y for each row in X such that Y ∈ {1, 2, ..., K}
    iteration ← 0
    converged ← false
    while (iteration < R ) AND (converged == false) do
        for centroid c ∈ C do
            Update location of centroid c using means of its cluster members
        for point p ∈ X do
            Assign p to its nearest centroid, update Y[p]
        iteration++
    return Y, C
```

samples in T such that $y \in \{$*eating and drinking, horizontal reaching*$\}$ are partitioned into a number of clusters where each cluster is a reduced representation of a "key pose". As can be seen from Figure 5.4, many trajectories follow some curved path. Therefore, our initial attempt was to use spectral clustering [103] to partition the trajectories. Spectral clustering is a clustering algorithm that performs dimensionality reduction by making use of the eigenvalues of the similarity matrix before applying traditional clustering algorithms such as K-means. There are three steps in perform spectral clustering:

1. Generate a similarity matrix S from the feature matrix T where s_{ij} is the similarity score between point i and j of T. The similarity measure can be: (1) a distance function such as the Euclidean distance; (2) a non-linear K-nearest-neighbour distance such that when a point j is among the K nearest neighbours of i, the similarity is the inverse of the distance, otherwise, the similarity is 0.

2. Compute the graph Laplacian matrix $L = I - D^{-1/2}SD^{-1/2}$ where $D = \sum_j S_{ij}$ and I is the identity matrix.

3. The eigenvectors of L are computed and the similarity matrix S is projected to the new basis defined by the new eigenvectors.

4. Clustering is performed by using the projected S such that non-Gaussian boundaries can be properly separated.

The intuition behind spectral clustering is that a similarity matrix can capture the relationship between the points even if the data have highly non-Gaussian boundaries. One major disadvantage of using spectral clustering is that the computational cost of computing the similarity matrix and eigenvectors is high even for a small scale problem. Therefore, we used the K-means algorithm [64] instead. K-means is a well-known clustering algorithm to partition the input data into K clusters, each of which is represented by the centroid of the data points in the cluster. It originates from the study of vector quantization [55]. For completeness, we briefly introduce K-means.

Algorithm 5.2 shows the pseudocode of K-means. The algorithm follows two steps, evaluating centroids by cluster means and updating cluster membership using the new centroids. On line 2 of Algorithm 5.2, the cluster label of each point is randomly initialized. Lines 5 to 10 are the

main loop that performs the evaluation-update steps. In each iteration, the centroids are updated by taking the means of the newly assigned cluster members (lines 6-7):

$$c^{r+1} = \frac{1}{|S^r|} \sum_{x_i \in S^r} x_i$$

where c^{r+1} is a centroid at $r+1^{th}$ iteration, S^r is the set of points assigned to the cluster in the previous iteration and x_i is a point with P features. The cluster memberships are updated by assigning points to the newly formed centroids using the nearest neighbour method (lines 8-9). This process is repeated until either no change in the cluster membership is detected or the maximum number of iterations is reached.

The benefit of using K-means is two fold. First, partitioning the data into disjoint regions represented by centroids reduces the effect of noise on the data. As can be seen in Figure 5.4(b), despite the actions being labelled as belonging to the same class, the variance between the way individuals perform the action causes the data to spread out across the space, which will yield poor recognition accuracy when used directly (Section 5.7.3). Giving a label using the cluster centroid serves the purpose of indexing, which tolerates imperfect matches in the training data as a small perturbation in the data sample will not cause a large change in the centroid location. Second, K-means is relatively fast and often requires very few iterations to converge when there is clustering structure in the data [64]. Consequently, adding K-means to the training stage will not significantly increase the running time, which is desirable.

Clustering results are shown in Figure 5.5 where the colour dots indicate cluster assignment of each point and the circled crosses are the the centroids. For each action in {*eating and drinking, horizontal reaching*}, we count how many *measurement samples* are in each of the clusters and we build a normalized histogram for all clusters (Figure 5.6). In Figure 5.6, it can be seen that {*horizontal reaching*} are distributed in clusters 4, 5, 6 and 7, whereas {*eating and drinking*} samples are distributed in clusters 1-4 and 8-10. The histogram values in clusters 5, 6 and 7 differ greatly for the two actions and intuitively, these clusters are the *key poses* that can be used to distinguish the actions. By saving the histogram to be used as templates for trained actions, new unknown actions can be identified.

The clustering in Figures 5.5 and 5.6 are performed with 10 clusters while Figure 5.6 shows

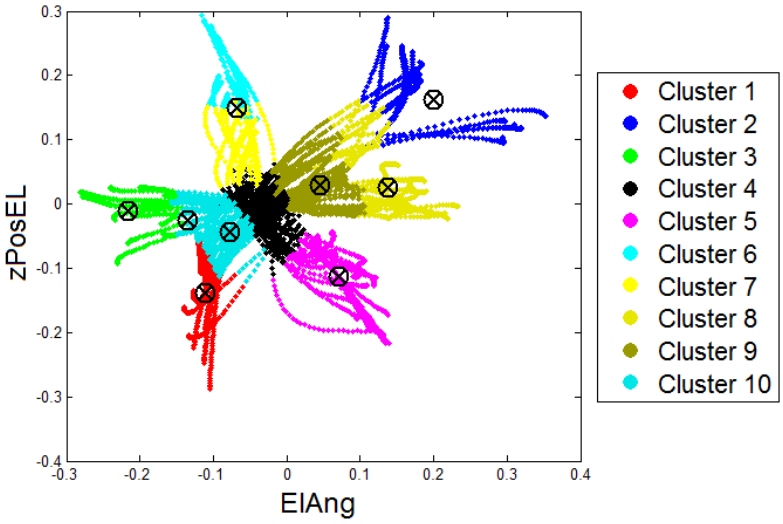
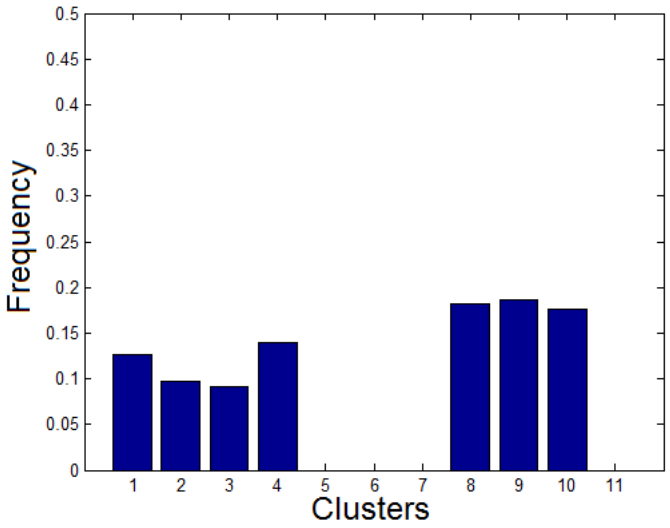


Figure 5.5: K-means clustering results of the trajectory features.

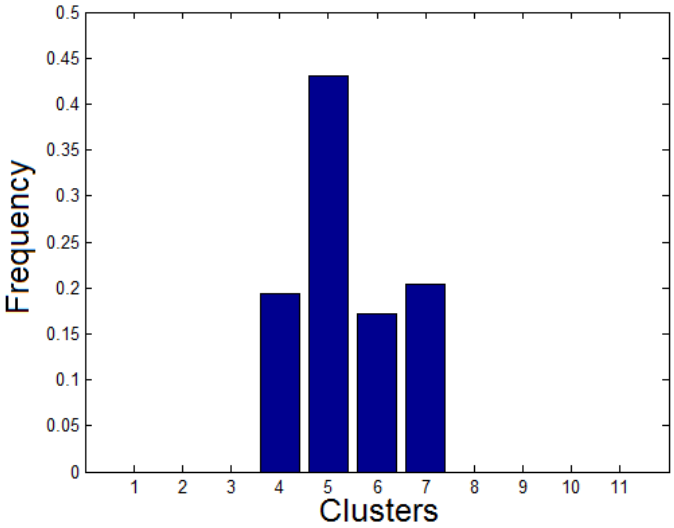
11 clusters. Cluster 11 is an additional "empty" cluster for the purpose of rejecting invalid actions. The value for cluster 11 value is set to 0 and the rejection process is explained in the next section. Since K-means requires the user to specify the number of clusters K, we investigate the impact of tuning parameter K on the performance of the algorithm in Section 5.7.3 where we show that the accuracy of our method is relatively insensitive to the number of clusters used.

5.6 Candidate Segmentation and Recognition

For new incoming data captured by the sensors, we recognise the action of the user with a two-step segmentation and recognition process. Recognizing actions in new data requires segmenting long continuous data streams. Two commonly used segmentation approaches are the sliding window approach [14] and the threshold-based approach [78]. In the sliding window approach, the data stream enters a buffer with constant length and is segmented when the buffer becomes full. After performing the recognition, the data in the buffer may either be deleted, which constitutes the non-overlapping window approach, otherwise, the newer part of the data is stored in the buffer to combine with new data streams, which is called the non-overlapping approach. The sliding window method has two disadvantages: (1) using overlaps requires additional computation; (2) the window size and percentage of overlap are difficult to determine. Threshold-based approaches,



(a) Eating and drinking



(b) Horizontal reaching

Figure 5.6: Histogram templates obtained using k-means clustering and trajectory features.

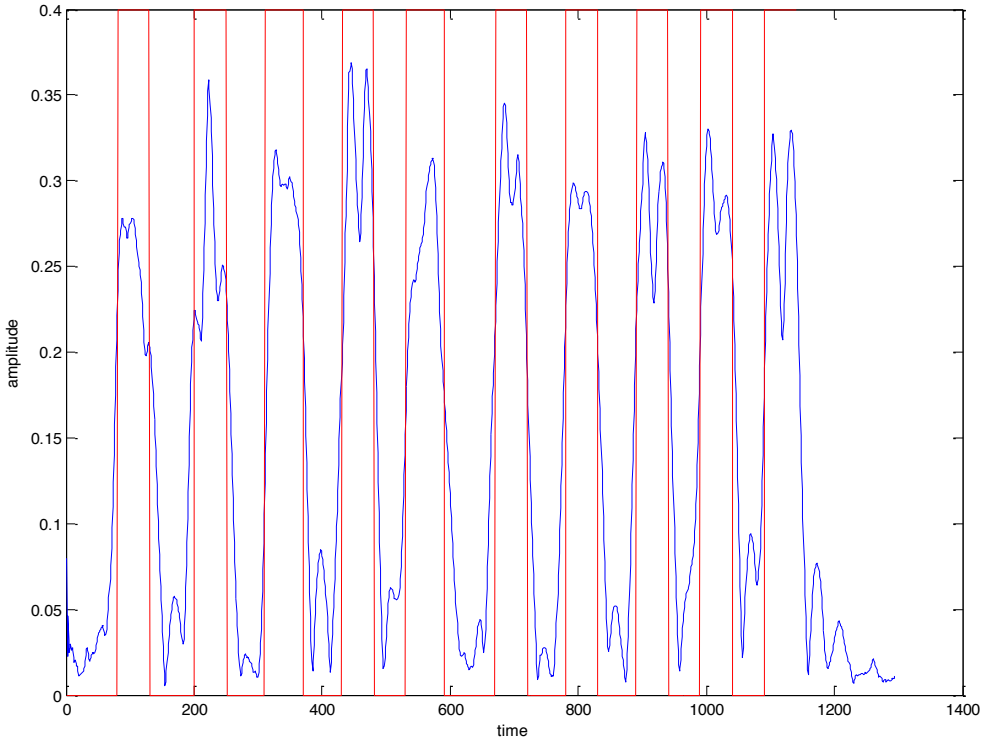


Figure 5.7: Segmentation of actions based on the magnitude of gyroscope signals

on the other hand, segment the data when the algorithm detects the existence of an action by using one or multiple thresholds. This approach is more efficient compared to the windowing method and is used in this work (Figure 5.7). We call a data segment that passes the threshold a *candidate*.

The first two *for* loops in Algorithm 5.3 illustrate our candidate segmentation approach. In the first pass (lines 2-4), samples whose sum of the square of the gyroscope x, y and z signals larger than a threshold g_{null} are labelled as valid samples. This is to identify the samples that correspond to active periods of the arm. In the second pass (lines 5-7), candidate samples with fewer neighbours than a minimum threshold are pruned to be invalid. This is to eliminate erroneous detections due to noise when the trajectory moves into an area of "empty space" with few training data points. Finally each block of contiguous samples is identified as a candidate to be classified.

Candidates are classified after being segmented (lines 8-18 in Algorithm 5.3). As the training algorithm generates a histogram template for each class of action H, the features of a test candidate are also transformed into a histogram h_{te}. This involves two steps: (1) assignment of each *measurement sample* with features defined in Table 5.3 to a cluster in Figure 5.5; and (2) counting

Algorithm 5.3: SegmentAndClassify

Data: $H \in \Re^{N\times(k_m+1)}$, $T \in \Re^{N\times P}$, $y \in \{null, action1, ..., actionN\}$, $Z \in \Re^{N\times P}$: Trajectory features of new data, $G_x, G_y, G_z \in \Re^{N\times 1}$: x,y,z axis gyro data

Result: $U \in \Re^{N\times 1}$, $U(i) \in y$: Predicted class labels, $V \in \Re^{N\times 1}$, $V \in \{1, ..., k_m + 1\}$: Cluster labels.

begin
  for $i \leftarrow 1$ *to* N **do**
    if $G_x(i)^2 + G_y(i)^2 + G_z(i)^2 > g_{null}$ **then**
      $W(i) \leftarrow TRUE, W \in \Re^{N\times 1}$
  forall the $W(i) = TRUE$ **do**
    if $W(i)$ *has* $<$ *w neighbours with value* $TRUE$ **then**
      $W(i) \leftarrow FALSE$
  while $\leftarrow 1, i \leq N$ **do**
    Scan W until finding $W(a:b) = TRUE$ and $W(a-1), W(b+1) = FALSE$
    Remove *mean* of each column of Z from $Z(a,:)$ to $Z(b,:)$
    for $j = a$ *to* b **do**
      Assign $V(j)$ to clusters $\{1, ..., k_m\}$ using KNN
      if *distance*$(Z, nearest\ cluster) > d_{max}$ **then**
        Assign $V(j)$ to cluster $k_m + 1$
    Build a histogram $h_{te} \in \Re^{1\times(k_m+1)}$ for $Z(a:b,:)$
    $U(a:b) \leftarrow$ argmin(distance(h_{te}, H))
    if *min*(*distance*$(h_{te}, H)) > s_{max}$ **then**
      $U(a:b) \leftarrow \{Null\}$
  return U, V

the cluster labels for all samples in a candidate to form the histogram h_{te}. For step 1, we considered two cluster assignment schemes: (1) assigning the label of the nearest *cluster centroid* to each test sample; (2) assigning the most frequently occurring label of R nearest *training samples* to each test sample. As the number of centroids is significantly smaller than the number of training samples, scheme 1 is less computationally expensive than scheme 2. However, since the clustering results of K-means is not unique, the positions of cluster centroids tend to vary greatly between runs and a large variance in the cluster assignment of test data can be observed when sheme 1 is implemented. On the other hand, despite an increase in the computational cost, scheme 2 performs reliably due to the fact that the most frequently occurring label in neighbourhood samples tends to be stable. Therefore, we implement scheme 2 in this work:

1. Compute the distance between the current *measurement sample* to all training samples and store into an array
2. Sort the distances into ascending order
3. Retrieve the top R samples and find the most frequently occurring label c of these samples
4. Set the label of the test sample to c

The K-nearest neighbour (KNN) algorithm is used to assign each *measurement sample* of the candidate to a cluster of the mean-centred trajectory features. There are two drawbacks of using KNN. Firstly, the time complexity of exact KNN assignment is $O(N^2log(N)P)$ where N is the number of samples in the candidate and P is the number of dimensions of each sample. For a large number of samples, the computational cost of brute-force KNN can be relatively high. In our case the number of samples is relatively small and each sample consists of four trajectory features only (ElAng, zPosEL, zPosWR, xyPosWR). Therefore, the computation time is tolerable. However, for larger numbers of samples and features, faster KNN implementations, such as KD trees and randomised K-means trees [100] may be used. The second drawback of KNN is that regardless of the actual distance, a cluster is always assigned to a sample. If the candidate belongs to an invalid action that is not contained in the training set, this will cause mis-classification errors. Therefore, we used a maximum distance threshold d_{max} to define the range of search. When the nearest cluster is further than d_{max}, the sample is assigned to the "empty cluster (cluster 11 in Figure 5.6), which

represents irrelevant actions. If a large number of samples in a candidate is assigned to this cluster, it will appear dissimilar to all of the actions trained and can be easily classified as invalid. The histogram h_{te} of a candidate is calculated by assigning all of its samples to the clusters, including the "empty" cluster, and then count the cluster labels. The distance between h_{te} and each histogram in H is computed, and the histogram h in H that has the minimum distance to h_{te} is found. If the distance between h and h_{te} is smaller than a threshold s_{max}, then the class of h is selected as the class of h_{te}. Otherwise, h_{te} is labelled as an invalid action and discarded.

5.7 Evaluation

5.7.1 Aim

In this section we evaluate the performance of action recognition using trajectory features and the proposed classifier. We compare trajectory features with raw data obtained directly from sensors, and we also compare the proposed classifier with a range of different benchmark classifiers. In addition, the increase in accuracy of post-processing the trajectory features by removing the mean of each action is also discussed.

5.7.2 Methodology

Data from two of the subjects are used as the training set, and testing is performed on the remaining two subjects. All the combinations of subjects have been exhaustively tested and results are averaged using the sample sizes as weights. Three classes of actions were used in the experiments: {*eating and drinking, reaching forward, invalid actions*}. The class {*invalid actions*} denotes irrelevant actions other than {*eating and drinking, reaching forward*} that the subjects perform naturally, such as picking up a fork and opening up the lunch box. The action classes are defined in this way to simulate the scenario of having a meal, which is an important daily activity. Correct recognition of these actions would provide important indications of the motor functions of the subject to physicians. The following well-known benchmark algorithms implemented in [28] were tested:

1. Linear discriminant analysis (LDA). A linear classifier that aims to find the projection hy-

perplane that minimizes the interclass variance and maximizes the distance between the projected means of the classes.

2. Quadratic discriminant analysis (QDA), like LDA but with quadratic kernels to find non-linear boundaries.

3. K-nearest Neighbours (KNN). An instance-based learning method which sets the class label to be the majority class of the nearest K points.

4. Nearest Centroid Classifier (NCC). Sets the class label to be the class with the nearest centroid.

Four tests were run to compare different combinations of features versus classifiers:

1. Trajectory features with the proposed classifier

2. Trajectory features with benchmark classifiers

3. Raw sensor data with the proposed classifier

4. Raw sensor data with benchmark classifiers

Weighted F scores are used to evaluate classification performance [124]. It is calculated as follows:

$$F = 2\left(\frac{precision * recall}{precision + recall}\right) \tag{5.1}$$

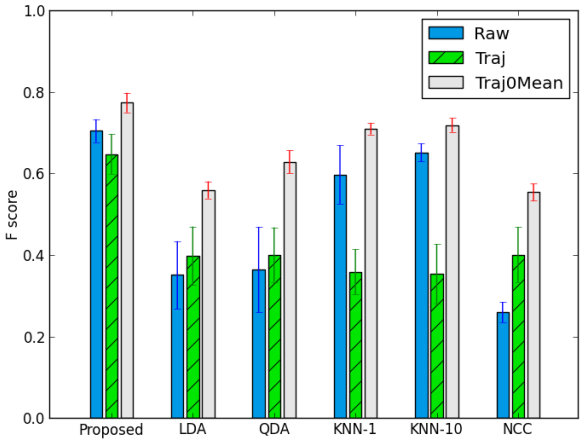
where

$$precision = \frac{recognized}{retrieved},\ recall = \frac{recognized}{relevant} \tag{5.2}$$

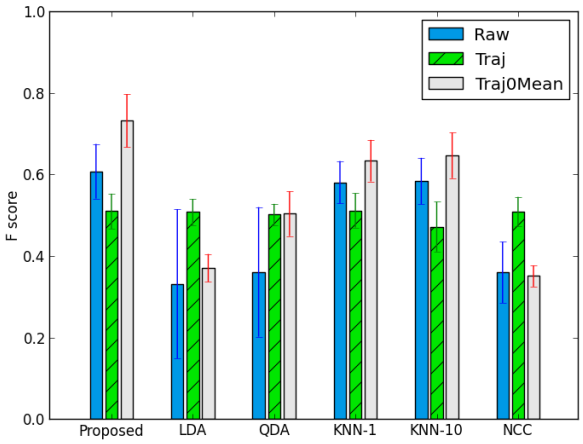
Since in real-life situations, invalid actions may occupy most of the signals captured by the sensors, we also evaluate the performance of rejection of these actions. We denote the $\{null\}$ class to be these actions, and aggregated F scores including and excluding the null class are calculated separately.

5.7.3 Results and Discussion

Figure 5.8 shows the results of the evaluation. It can be seen that trajectory features are superior to raw sensor data in the task of action recognition. When trajectory features are used, the



(a) Including Null class



(b) Excluding Null class

Figure 5.8: F scores of the proposed algorithm and benchmark algorithms with 1) raw sensor data (Raw); 2) trajectory features (Traj); and 3) mean-centred trajectory features (Traj0Mean).

overall recognition accuracy improves from raw sensor data for all the algorithms tested. The best performing algorithm is the proposed algorithm while the best amongst the benchmarks is KNN. While QDA scores badly with raw sensor data, the largest improvement in F scores is observed when switching to the trajctory features (an increase from 0.34 to 0.60 including null class and 0.36 to 0.50 excluding null class). For most classifiers, mean-centred trajectory features yield significantly better results than un-centred trajectories. This suggests that mean-centring is indeed necessary in recognising the actions. Strangely, mean-centred trajectory features are effective for LDA and NCC when evaluating cases including the Null class but not with cases excluding the Null class. This may be caused by the tendency of overfitting the data towards the Null class by some models and care should be taken when using classification algorithms.

The proposed classifier out-performs all benchmark algorithms with both trajectory features (mean centred) and raw sensor data, which shows its capability to be used as an action classifier (Figure 5.8). The proposed algorithm achieves an F score of 0.705 ± 0.028 for classification including the null class and 0.608 ± 0.067 excluding null class. The K-nearest neighbour method (KNN) is the best classifier among the benchmark algorithms with weighted F scores of 0.652 $\pm$ 0.022 (including null class) and 0.584 ± 0.057 (excluding null class) when raw sensor data are used. It achieves an accuracy slightly inferior to the proposed algorithm under trajectory features. The proposed algorithm differs from KNN by a k-means clustering step and a threshold filter, which explains the similarity in performance. Recognition accuracy of other classifiers are significantly worse with raw sensor data as well as trajectory features.

These results show that the high level trajectory features improve recognition accuracy for detecting and classifying eating, drinking and horizontal reaching tasks. Note that selecting a different number of clusters can affect recognition accuracy (Figure 5.9), but the accuracy remains approximately constant when the number of clusters is set between 10 and 18. This shows the robustness of the proposed classifier against parameter variations.

5.8 Conclusions and Future Work

We have shown that using trajectory features can improve the recognition accuracy for $\{$*eating or drinking, horizontal reaching*$\}$ actions compared with using raw sensor data. Furthermore, our

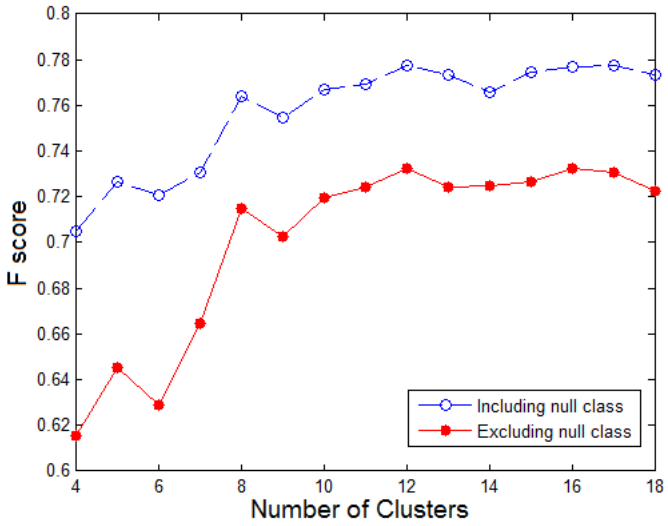


Figure 5.9: F scores for different number of clusters in k-means.

proposed clustering-based classifier out-performs all benchmark classifiers tested with F scores of 0.705 when using raw data and 0.774 using trajectory features. It can be seen that trajectory features are superior to raw data in recognizing these actions. Although improvements can be made, we argue that our method shows the potential of trajectory features in action recognition and we expect it to be useful in a wide range of applications such as stroke rehabilitation and sports.

There exists a number of possible future work directions.

1. We assumed a supervised learning set-up where all the actions have been labelled, and the training process of our algorithm requires actions of the same class to be performed repeatedly. However, in real-life situations, a user might become frustrated if the algorithm asks for too many repetitions of the same action. In this work, the model was trained with 10 instances in a single class and a possible direction to investigate is the feasibility of reducing the number of training instances (e.g., to five or one) while still maintaining a high classification accuracy. A related direction for research is to design an active learning approach, that monitors the training data that has been collected from users and decides when sufficient data has been collected. This may require the development of a suitable cluster validity measure for assessing the quality of the trajectory clusters.

2. We performed the activity segmentation with a simple threshold algorithm as in [73], since

the focus of this work is feature extraction and classifier design. This approach is suitable in clinical or sports research applications when an expert is often present when the subject performs the action and immediate interruption can be given if the segmentation is not performed correctly. However, in daily-life situations when patients continuously monitor their own progress, incorrect segmentation will be costly in terms of classification error and better algorithms should be investigated. Methods that exploit temporal information and combine segmentation and classification into a single system can be a direction to explore (such as Hidden Markov Models).

3. We compared the proposed algorithm with a number of benchmark algorithms, and we achieved higher accuracy than all of the benchmarks with trajectory features. However, we did not compare our classifier with state-of-the-art classification algorithms used in action recognition, which could be a direction for future research.

4. In this work, we only investigate classification of three classes, $\{$*eating or drinking, horizontal reaching, null class*$\}$. However, subjects may perform many more actions in daily life, which are not available from the training set. A possible research direction is to include more classes of upper limb activities into the system without significantly increase the training cost. Although a completely unsupervised approach may be infeasible, semi-supervised learning may be useful in this scenario where labelled data from other subjects are combined with unlabelled data from the current subject to perform the classification.

5. In this work, we captured data from four subjects to perform the evaluation. For large scale deployment of the system, it is necessary to recruit more subjects to evaluate the algorithm, which can be time-consuming and expensive. Although in some past studies, more than four subjects were recruited in the evaluation [14, 80, 127], other authors have also performed a proof-of-concept evaluation with no more than four subjects [28, 94, 116, 120]. A future research direction would be to evaluate our system with more subjects.

Chapter 6

Conclusions and Future Work

6.1 Summary of Contributions

THIS thesis has addressed the issue of trajectory mining in the context of the Internet of Things. There are a growing number of devices that are equipped with sensors that can generate spatio-temporal traces, i.e., *trajectories*. With the infrastructure of the Internet of Things (IoT), mining and analytics of trajectory data will benefit a number of applications. In particular, we have focused on three application scenarios, namely social, transport and healthcare.

In Chapter 3, we studied trip recommendation for tourists, and we formulated the problem as an Orienteering Problem with multiple objectives. We focused on two main challenges: (1) the objectives, i.e., user interest, popularity of the points-of-interest (POIs) and crowdedness of the POIs, may be in conflict and no global best solution may be found; (2) check-in data used by existing trip recommendation studies are sparse, which are not suitable to be used to optimise time-dependent objectives such as the crowdedness of a POI. We identified that trajectory data submitted by Internet users can be combined with anonymous pedestrian flow data in trip recommendation, and the use of pedestrian sensing data overcomes the data sparsity. We solved the multi-objective time-dependent Orienteering Problem by devising a form of the Ant Colony Optimisation algorithm to account for the time-dependency. Although trip recommendation is a well-researched field, to the best of our knowledge, the combination of user trajectories with pedestrian sensor information to recommend interesting yet less crowded trips is novel. We gathered user trajectory data from Flickr and pedestrian volume information in the City of Melbourne from a number of pedestrian counters. The user interest was estimated using the Flickr trajectories with a user-based collaborative filtering algorithm. POI popularity was calculated based on the number of times the POIs have

been visited when user trajectories were aggregated for each POI to produce a normalised score. A comparison with state-of-the-art trip recommendation algorithms shows that our method can achieve an appropriate balance in the objectives. A case study using real trip data has shown that the proposed system that combines trajectory information with IoT sensor data can allow better trips to be found with reduced crowdedness and high user interest.

In Chapter 4, we focused on using trajectory mining for transport applications. Specifically, we studied the identification of the differences in traffic flows before and after an event (e.g., a road closure) and the impact of the event on the nearby traffic. We utilised contrast mining techniques to highlight subgraphs formed by road segments that have large changes in traffic flows. We also identified parts of the road network that frequently overload due to the event. To the best of our knowledge, little work has focused on mining the differences of traffic flows under the impact of a special event, and the use of contrast mining techniques in vehicle trajectory mining is novel. We treated road segments as sets of edges in a graph and compare multiple road segment sets simultaneously using a notion of *growth rate*. The *growth rate* is a score that computes the change in the *support* of the sets, and a high *growth rate* will indicate a large change in traffic volume. We ranked the sets by their *growth rate* and found road segments that have significant increase and decrease of traffic after the event. This information may provide the authorities and motorists with helpful insights in planning and decision making as well as the understanding of the impact of events on the traffic.

In Chapter 5, we focused on healthcare applications and extracted motion trajectories from wearable devices to perform upper limb action recognition for the wearer. This system is motivated by the need of patient monitoring in the home environment, especially for physically impaired individuals who are undergoing rehabilitation from conditions like stroke. As the users may perform the activities without any form of supervision in the home environment, the variations between the same class of activities may reduce recognition accuracy, which is the main challenge of the problem. To solve the problem, we recorded the motion data using accelerometers, gyroscopes and magnetometers and reconstructed the upper limb movement trajectories with an arm model and a sensor fusion algorithm. We used the K-Means clustering algorithm to cluster the trajectories such that each cluster acts as a representative snapshot of the motion. We generated histograms of the clusters for each action class, and consequently transformed the trajectory fea-

tures into a new space spanned by the cluster histograms. The new actions were then recognised using the histogram features by (1) matching each trajectory sample with the clustering using K-nearest neighbours and (2) generating a histogram of its own by counting the points in each cluster. Although there has been a large collection of literature on action recognition, to the best of our knowledge, the transformation of sensor data into trajectories and the generation of representative motion snapshots with these trajectories in action recognition is novel. We compared our algorithm with a number of benchmarks and the results were promising in the way that we have improved recognition accuracy without compromising significantly on time complexity.

6.2 Future Research

We now present a number of broad future research directions based on the problems studied in this thesis. For specific research problems related to the method proposed in each chapter, we have included some detailed directions in the summaries of Chapters 3, 4 and 5.

For the trip recommendation problem, we assumed that the mode of transport was walking. This is an appropriate assumption for small tourism cities like Melbourne but maybe unrealistic for mega-cities like Tokyo. A real-life trip recommendation system may need to consider the impact of mode of transportation on the travel time, which can be extremely complicated. A complete trip may involve several transport modes such as trains, trams, buses, taxis and walking. The transit time between different transport modes will become a variable which must be estimated. Due to the uncertainty in external factors such as weather conditions, accurate estimation of transit time is going to be one of the main challenges. To provide a realistic estimate, the first step is to incorporate the public transport timetable into the system, which requires collaboration from various parties and the combination of a number of data sources. In real-life situations, timetables may be subject to changes and there may be a delay between the update of the data source and the actual timetable. The identification of changes in the public transport timetable may be another main challenge and a future research direction.

The second direction is the possible integration of the trip recommendation and road traffic analysis system. As can be seen in the trip recommendation problem, the travel time plays an important role in the algorithmic generation of trips. An event that has large impact on the traffic may

distort travel time estimation significantly, and not all events can be known prior to its occurrence. Tackling this problem is likely to require the immediate feedback of real-time quantitative traffic information captured from various road sensors to users' navigation devices. Currently, drivers may be able to obtain qualitative information on road conditions, such as which roads are affected by accidents, but not quantitative information, i.e., to what extent does the disturbance spread, which travel paths are affected and how much time will be lost. The estimation of this information using IoT sensors will be a challenging but rewarding problem. If such information is available to ordinary users in the future, trip recommendation may also benefit from it and the quality of the recommended trips may be further improved.

For action recognition, we focused solely on recognising upper limb actions due to our sensor setup and the objective of patient monitoring at home. A more general-purpose action recognition system that uses a mobile phone and a smart watch may be of interest to the general public. Due to the variations in the placement locations of the phone (e.g., hip pocket or thigh pocket) and the watch, an arm model may be unavailable and trajectory features may be difficult to extract. Feature extraction from the combination of these two devices may be an important problem. Another problem of interest may be the incorporation of action recognition with location recommendation. For example, when the system detects the *running* activity in a park, it automatically discovers shops that sell drinks when running is over and the user unlocks the phone. The correct discovery of the activity by merging multiple sources of data and the intention after completing the activity will be a challenge. In addition, user privacy protection will be another problem as some users may consider the recommendation as intrusive. The determination of an appropriate time for the recommendation may be another interesting direction.

The above three directions have naturally lead to the fourth question: how do we properly integrate trajectories captured from different objects, sensors and transport modes? Due to practical issues such as battery life, trajectories recorded by different devices tend to have varying sampling frequencies. Working on data with different granularities usually requires some form of aggregation to be performed on the data. Methods currently deployed are: (1) grid-based, which is simply dividing space into grid cells and aggregate trajectories within each cell; or (2) area-based, i.e., aggregation using some form of natural boundaries, such as roads on the map. Both of these methods suffer from the problem of arbitrariness. For example, what is the appropriate size of a

grid cell on a map? Which roads should be used to draw the boundaries? The choice of these parameters can be somewhat subjective. An alternative method to investigate is trajectory clustering, i.e., aggregate and divide the trajectories using the structures observed within the trajectories. If a good clustering algorithm is used, the solution may be more stable than using arbitrary grid cells. In this way, we could devise a more comprehensive platform for trajectory analysis based on fusing heterogeneous data sources from the Internet of Things.

In this thesis, we have presented novel theoretical contributions to contrast mining of trajectories, extracting trajectory features for classification, and using trajectory information to guide optimisation problems in decision making. While this thesis has been presented from the perspective of three application challenges for trajectory mining, we can also consider how the novel theoretical contributions of our research complement each other and motivates future research from a theoretical perspective. We highlight two such cases of potential future theoretical research.

In the first case, contrast mining of trajectories and trajectory classification have the potential to be complementary techniques. One may use contrast mining of trajectories to extract significant discriminating trajectory features that can be used as features for action recognition, rather than relying on the raw trajectory features alone. For example, in the arm action classification problem in Chapter 5, rather than identifying (static) key poses, we could use trajectory contrast mining to find key sub-trajectories that discriminate between different actions. Similarly, in Chapter 3, we may be able to use the contrast trajectories that have been discovered from traffic flows to help in learning a classifier that can infer the likely cause of a traffic incident, such as an accident causing a partial road closure or a special sports event causing congestion.

In the second case, we can investigate the potential for speed-up learning in optimisation problems, such as the orienteering problem, based on frequently occurring sub-trajectories that have been observed in previous solutions to the problem. By finding frequently occurring sub-trajectories in the tours generated from past solutions of the problem, it may be possible to use these sub-trajectories to "seed" the search for new tours. A key theoretical challenge in this context is how to identify sub-trajectories with high utility from past problems that are highly likely to be relevant to future problems. Rather than remembering all solutions to all past problems, identifying these sub-trajectories may save both time and space, and it may be possible to improve the efficiency of search for problems such as the Orienteering Problem proposed in Chapter 3.

These are just two examples of how the theoretical contributions in this thesis could provide the foundation for future research into trajectory mining for the Internet of Things.

Bibliography

[1] "Melbourne landmarks and places of interest," 2015, https://data.melbourne.vic.gov.au/Assets-Infrastructure/Landmarks-and-Places-of-Interest/j5vt-ppat, Accessed: 2015-10-20.

[2] "Melbourne pedestrian data," 2015, http://www.pedestrian.melbourne.vic.gov.au, Accessed: 2015-10-20.

[3] C. C. Aggarwal, "Outlier analysis," in *Data Mining*. Springer, 2015, pp. 237–263.

[4] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[5] K. Altun and B. Barshan, "Human activity recognition using inertial/magnetic sensor units," in *International Workshop on Human Behavior Understanding*. Springer, 2010, pp. 38–51.

[6] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognition*, vol. 43, no. 10, pp. 3605–3620, 2010.

[7] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *International Workshop on Ambient Assisted Living*. Springer, 2012, pp. 216–223.

[8] R. Arnott, A. De Palma, and R. Lindsey, "Does providing information to drivers reduce traffic congestion?" *Transportation Research Part A: General*, vol. 25, no. 5, pp. 309–318, 1991.

[9] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[10] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga, "Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey," in *23rd International Conference on Architecture of Computing Systems (ARCS)*. VDE, 2010, pp. 1–10.

[11] J. Bailey, T. Manoukian, and K. Ramamohanarao, "Fast algorithms for mining emerging patterns," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2002, pp. 39–50.

[12] F. Banaei-Kashani, C. Shahabi, and B. Pan, "Discovering patterns in traffic sensor data," in *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on GeoStreaming*. ACM, 2011, pp. 10–16.

[13] N. Bansal, A. Blum, S. Chawla, and A. Meyerson, "Approximation algorithms for deadline-tsp and vehicle routing with time-windows," in *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*. ACM, 2004, pp. 166–174.

[14] L. Bao and S. S. Intille, *Activity recognition from user-annotated acceleration data*. Springer, 2004, vol. 3001, pp. 1–17.

[15] S. D. Bay and M. J. Pazzani, "Detecting change in categorical data: Mining contrast sets," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1999, pp. 302–306.

[16] R. J. Bayardo Jr, "Efficiently mining long patterns from databases," *ACM Sigmod Record*, vol. 27, no. 2, pp. 85–93, 1998.

[17] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[18] B. Berjani and T. Strufe, "A recommendation system for spots in location-based online social networks," in *Proceedings of the Fourth Workshop on Social Network Systems*. ACM, 2011, p. 4.

[19] N. Bicocchi, M. Mamei, and F. Zambonelli, "Detecting activities from body-worn accelerometers via instance-based algorithms," *Pervasive and Mobile Computing*, vol. 6, no. 4, pp. 482–495, 2010.

[20] D. Billsus, C. A. Brunk, C. Evans, B. Gladish, and M. Pazzani, "Adaptive interfaces for ubiquitous web access," *Communications of the ACM*, vol. 45, no. 5, pp. 34–38, 2002.

[21] F. Bohnert, I. Zukerman, and J. Laures, "Geckommender: Personalised theme and tour recommendations for museums," in *User Modeling, Adaptation, and Personalization*. Springer, 2012, pp. 26–37.

[22] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[23] I. Brilhante, J. A. Macedo, F. M. Nardini, R. Perego, and C. Renso, "Where shall we go today?: planning touristic tours with tripbuilder," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2013, pp. 757–762.

[24] P. S. Castro, D. Zhang, and S. Li, "Urban traffic modelling and prediction using large scale taxi gps traces," in *International Conference on Pervasive Computing*. Springer, 2012, pp. 57–72.

[25] M. Ceci, A. Appice, and D. Malerba, "Discovering emerging patterns in spatial databases: A multi-relational approach," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2007, pp. 390–397.

[26] R. Chan, Q. Yang, and Y.-D. Shen, "Mining high utility itemsets," in *Third IEEE International Conference on Data Mining*. IEEE, 2003, pp. 19–26.

[27] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[28] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Trster, J. d. R. Milln, and D. Roggen, "The opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2033–2042, 2013.

[29] S. Chawla, Y. Zheng, and J. Hu, "Inferring the root cause in road traffic anomalies," in *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012, pp. 141–150.

[30] C. Chekuri and M. Pal, "A recursive greedy algorithm for walks in directed graphs," in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. IEEE, 2005, pp. 245–253.

[31] C. Chen, D. Zhang, B. Guo, X. Ma, G. Pan, and Z. Wu, "Tripplanner: Personalized trip planning leveraging heterogeneous crowdsourced digital footprints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1259–1273, 2015.

[32] K. Chen and S. Har-Peled, "The orienteering problem in the plane revisited," in *Proceedings of the 22nd Annual Symposium on Computational Geometry*. ACM, 2006, pp. 247–254.

[33] C. Cheng, H. Yang, M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation." in *IJCAI*, vol. 13, 2013, pp. 2605–2611.

[34] C.-H. Chu, W.-C. Wu, C.-C. Wang, T.-S. Chen, and J.-J. Chen, "Friend recommendation for location-based mobile social networks," in *Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. IEEE, 2013, pp. 365–370.

[35] R. R. Coifman and M. V. Wickerhauser, "Entropy-based algorithms for best basis selection," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 713–718, 1992.

[36] S. Counts and M. Smith, "Where were we: communities for sharing space-time trails," in *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*. ACM, 2007, p. 10.

[37] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu, "Automatic construction of travel itineraries using social breadcrumbs," in *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*. ACM, 2010, pp. 35–44.

[38] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1–18, 2000.

[39] G. Demiris and B. K. Hensel, "Technologies for an aging society: a systematic review of smart home applications," *Yearbook of Medical Informatics*, vol. 3, pp. 33–40, 2008.

[40] G. Demiris, B. K. Hensel, M. Skubic, and M. Rantz, "Senior residents perceived need of and preferences for smart home sensor technologies," *International Journal of Technology Assessment in Health Care*, vol. 24, no. 01, pp. 120–124, 2008.

[41] S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas, and D. J. Cook, "Simple and complex activity recognition through smart phones," in *8th International Conference on Intelligent Environments (IE)*. IEEE, 2012, pp. 214–221.

[42] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[43] W. Ding, T. F. Stepinski, and J. Salazar, "Discovery of geospatial discriminating patterns from remote sensing datasets." in *SDM*. SIAM, 2009, pp. 425–436.

[44] W. J. Dixon and F. J. Massey Jr, "Introduction to statistical analysis ." 1957.

[45] M. C. Domingo, "An overview of the internet of things for people with disabilities," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 584–596, 2012.

[46] G. Dong and J. Bailey, *Contrast Data Mining: Concepts, Algorithms, and Applications*. CRC Press, 2012.

[47] G. Dong and J. Li, "Efficient mining of emerging patterns: Discovering trends and differences," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1999, pp. 43–52.

[48] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 36–39.

[49] H. Fan and K. Ramamohanarao, "Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 6, pp. 721–737, 2006.

[50] F. Folianto, Y. S. Low, and W. L. Yeow, "Smartbin: Smart waste management system," in *IEEE 10th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. IEEE, 2015, pp. 1–2.

[51] Forbes, "Beijing traffic jam," 2010, http://www.forbes.com/sites/jimgorzelany/2015/10/15/the-worlds-worst-traffic-jams-ever/76f3f1d36c6c, Accessed: 2016-9-27.

[52] A. Gallagher, D. Joshi, J. Yu, and J. Luo, "Geo-location inference from image content and user tags," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2009, pp. 55–62.

[53] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems," *Journal of Heuristics*, vol. 20, no. 3, pp. 291–328, 2014.

[54] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and N. Vathis, "Heuristics for the time dependent team orienteering problem: Application to tourist route planning," *Computers and Operations Research*, vol. 62, pp. 36 – 50, 2015.

[55] A. Gersho and R. M. Gray, *Vector quantization and signal compression*. Springer Science & Business Media, 2012, vol. 159.

[56] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics (NRL)*, vol. 34, no. 3, pp. 307–318, 1987.

[57] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[58] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag, "Adaptive fastest path computation on a road network: a traffic mining approach," in *Proceedings of the 33rd International Conference on Very Large Data Bases*. VLDB Endowment, 2007, pp. 794–805.

[59] U. Guardian, "Shanghai: dozens killed and injured in stampede at new year celebrations," https://www.theguardian.com/world/2014/dec/31/shanghai-35-people-killed-42-injured-new-year-crush.

[60] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[61] A. Gunawan, H. C. Lau, and Z. Yuan, *A Mathematical Model and Metaheuristics for Time Dependent Orienteering Problem*. Helmut-Schmidt-Univ., Univ. der Bundeswehr Hamburg, 2014.

[62] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[63] Y. Hao and R. Foster, "Wireless body sensor networks for health-monitoring applications," *Physiological Measurement*, vol. 29, no. 11, p. R27, 2008.

[64] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[65] D. M. Hawkins, *Identification of outliers*. Springer, 1980, vol. 11.

[66] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[67] S. Hodges, L. Williams, E. Berry, S. Izadi, J. Srinivasan, A. Butler, G. Smyth, N. Kapur, and K. Wood, "Sensecam: A retrospective memory aid," in *International Conference on Ubiquitous Computing*. Springer, 2006, pp. 177–193.

[68] S. Hussain, S. Schaffner, and D. Moseychuck, "Applications of wireless sensor networks and rfid in a smart home environment," in *Seventh Communication Networks and Services Research Conference*. IEEE, 2009, pp. 153–157.

[69] S. Ichoua, M. Gendreau, and J.-Y. Potvin, "Vehicle dispatching with time-dependent travel times," *European Journal of Operational Research*, vol. 144, no. 2, pp. 379–396, 2003.

[70] S. Jiang, X. Qian, T. Mei, and Y. Fu, "Personalized travel sequence recommendation on multi-source big social media," *IEEE Transactions on Big Data*, vol. PP, no. 99, pp. 1–1, 2016.

[71] T. Joachims, "Making large scale svm learning practical," Universität Dortmund, Tech. Rep., 1999.

[72] A. Juels, "Rfid security and privacy: A research survey," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 381–394, 2006.

[73] H. Junker, O. Amft, P. Lukowicz, and G. Trster, "Gesture spotting with body-worn inertial sensors to detect user activities," *Pattern Recognition*, vol. 41, no. 6, pp. 2010–2024, 2008.

[74] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 2, pp. 108–118, 2000.

[75] C. Khatri and K. Mardia, "The von mises-fisher matrix distribution in orientation statistics," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 95–106, 1977.

[76] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[77] R. Kotagiri, S. Karunasekera, L. Kulik, E. Tanin, R. Zhang, H. Xie, and E. B. Khunayn, "Smarts: Scalable microscopic adaptive road traffic simulator," *ACM Transactions on Intelligent Systems and Technology (TIST) (to appear)*, 2016.

[78] N. C. Krishnan, P. Lade, and S. Panchanathan, "Activity gesture spotting using a threshold model based on adaptive boosting," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2010, pp. 155–160.

[79] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura, "Travel route recommendation using geotags in photo sharing sites," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. ACM, 2010, pp. 579–588.

[80] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[81] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[82] B.-H. Lee, H.-N. Kim, J.-G. Jung, and G.-S. Jo, "Location-based service with context data for a restaurant recommendation," in *International Conference on Database and Expert Systems Applications*. Springer, 2006, pp. 430–438.

[83] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *IEEE 24th International Conference on Data Engineering*. IEEE, 2008, pp. 140–149.

[84] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. ACM, 2007, pp. 593–604.

[85] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma, "Mining user similarity based on location history," in *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2008, p. 34.

[86] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[87] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, "Personalized tour recommendation based on user interests and points of interest visit durations," in *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 1778–1784.

[88] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[89] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uwave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 657–675, 2009.

[90] S. Liu, L. M. Ni, and R. Krishnan, "Fraud detection from taxis' driving behaviors," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 1, pp. 464–472, 2014.

[91] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing, "Discovering spatio-temporal causal interactions in traffic data streams," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 1010–1018.

[92] E. H.-C. Lu, C.-Y. Chen, and V. S. Tseng, "Personalized trip recommendation with multiple constraints by mining user check-in behaviors," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 2012, pp. 209–218.

[93] H. Luinge, P. Veltink, and C. Baten, "Ambulatory measurement of arm orientation," *Journal of Biomechanics*, vol. 40, no. 1, pp. 78–85, 2007.

[94] P. Lukowicz, J. A. Ward, H. Junker, M. Stger, G. Trster, A. Atrash, and T. Starner, *Recognizing workshop activity using body worn microphones and accelerometers*. Springer, 2004, vol. 3001, pp. 18–32.

[95] A. Milenković, C. Otto, and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation," *Computer communications*, vol. 29, no. 13, pp. 2521–2533, 2006.

[96] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl, "Movielens unplugged: experiences with an occasionally connected recommender system," in *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 2003, pp. 263–266.

[97] M. Miller and C. Gupta, "Mining traffic incidents to forecast impact," in *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*. ACM, 2012, pp. 33–40.

[98] MSRA, "T-drive," 2010, https://www.microsoft.com/en-us/research/project/t-drive-driving-directions-based-on-taxi-traces/.

[99] ——, "Geolife project," 2012, https://www.microsoft.com/en-us/download/details.aspx?id= 52367.

[100] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." *VISAPP (1)*, vol. 2, no. 331-340, p. 2, 2009.

[101] V. Nagarajan and R. Ravi, "The directed orienteering problem," *Algorithmica*, vol. 60, no. 4, pp. 1017–1030, 2011.

[102] B. Nath, F. Reynolds, and R. Want, "Rfid technology and applications," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 22–24, 2006.

[103] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[104] J. Nichols and B. A. Myers, "Controlling home and office appliances with smart phones," *IEEE Pervasive Computing*, vol. 5, no. 3, pp. 60–67, 2006.

[105] X.-X. Niu and C. Y. Suen, "A novel hybrid cnn–svm classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318–1325, 2012.

[106] OpenStreetMap, "Open street map," 2016, https://www.openstreetmap.org/.

[107] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi, "Crowd sensing of traffic anomalies based on human mobility and social media," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 344–353.

[108] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng, "On mining anomalous patterns in road traffic streams," in *International Conference on Advanced Data Mining and Applications*. Springer, 2011, pp. 237–251.

[109] ——, "On detection of emerging anomalous traffic patterns using gps data," *Data & Knowledge Engineering*, vol. 87, pp. 357–373, 2013.

[110] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 1, pp. 1–12, 2010.

[111] S. Patel, R. Hughes, T. Hester, J. Stein, M. Akay, J. G. Dy, and P. Bonato, "A novel approach to monitor rehabilitation outcomes in stroke survivors using wearable technology," *Proceedings of the IEEE*, vol. 98, no. 3, pp. 450–461, 2010.

[112] H. Qi and J. B. Moore, "Direct kalman filtering approach for gps/ins integration," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 2, pp. 687–693, 2002.

[113] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[114] L. Rabiner and B. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[115] P. Rashidi and D. J. Cook, "Com: A method for mining and monitoring human activity patterns in home-based health monitoring systems," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, p. 64, 2013.

[116] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *AAAI*, 2005, pp. 1541–1546.

[117] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[118] A. M. Sabatini, "Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 7, pp. 1346–1356, 2006.

[119] S. Salcedo-Sanz, D. Manjarres, Á. Pastor-Sánchez, J. Del Ser, J. A. Portilla-Figueras, and S. Gil-Lopez, "One-way urban traffic reconfiguration using a multi-objective harmony search approach," *Expert Systems with Applications*, vol. 40, no. 9, pp. 3341–3350, 2013.

[120] Z. Sen, J. M. H. Ang, X. Wendong, and T. Chen Khong, "Detection of activities by wireless sensors for daily life surveillance: Eating and drinking," *Sensors*, vol. 9, no. 3, pp. 1499–1517, 2009.

[121] J. Shang, Y. Zheng, W. Tong, E. Chang, and Y. Yu, "Inferring gas consumption and pollution emission of vehicles throughout a city," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 1027–1036.

[122] R. W. Sinnott, "Virtues of the Haversine," *Sky and Telescope*, vol. 68, p. 158, Dec. 1984.

[123] Sohu, "Beijing traffic jam," 2016, http://news.sohu.com/20100819/n274312074.shtml, Accessed: 2016-9-27.

[124] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2006, pp. 1015–1021.

[125] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. M. S. De Souza, and V. Trifa, "Soa-based integration of the internet of things in enterprise services," in *IEEE International Conference on Web Services*. IEEE, 2009, pp. 968–975.

[126] T. F. Stepinski, J. Salazar, and W. Ding, "Discovering spatio-social motifs of electoral support using discriminative pattern mining," in *Proceedings of the First International Conference and Exhibition on Computing for Geospatial Research & Application*. ACM, 2010, p. 39.

[127] L. Sun, D. Zhang, B. Li, B. Guo, and S. Li, "Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations," in *International Conference on Ubiquitous Intelligence and Computing*. Springer, 2010, pp. 548–562.

[128] S. Suvorova, S. Howard, and B. Moran, "Bayesian recursive estimation on the rotation group," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6411–6415.

[129] A. Takizawa, K. Yoshida, and N. Katoh, "Applying graph mining to discover substructures of room layouts which affect the rent of apartments," in *IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2007, pp. 3512–3518.

[130] A. Takizawa, "Classification and feature extraction of criminal occurrence points using caep with transductive clustering," *Procedia-Social and Behavioral Sciences*, vol. 21, pp. 83–92, 2011.

[131] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "The new data and new challenges in multimedia research," *arXiv preprint arXiv:1503.01817*, 2015.

[132] W. R. Tobler, "A computer movie simulating urban growth in the detroit region," *Economic Geography*, vol. 46, pp. 234–240, 1970.

[133] T. Tsiligirides, "Heuristic methods applied to orienteering," *Journal of the Operational Research Society*, pp. 797–809, 1984.

[134] A. Tuzhilin, "Towards the next generation of recommender systems," in *Proceedings of the First International Conference on E-Business Intelligence (ICEBI2010),*. Atlantis Press, 2010.

[135] US, "Us census public-use microdata samples (pums)," 1990, https://www.census.gov/main/ www/pums.html, Accessed: 2016-09-12.

[136] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden, "The city trip planner: An expert system for tourists," *Expert Systems with Applications*, vol. 38, no. 6, pp. 6540 – 6546, 2011.

[137] C. Verbeeck, K. Srensen, E.-H. Aghezzaf, and P. Vansteenwegen, "A fast solution method for the time-dependent orienteering problem," *European Journal of Operational Research*, vol. 236, no. 2, pp. 419 – 432, 2014.

[138] VicRoads, "Melbourne traffic data," 2014, http://www.theage.com.au/victoria/melbournes-long-hard-road-ahead-freeway-traffic-on-course-to-double-in-20-years-20160219-gmyhv1.html, Accessed: 2016-9-27.

[139] L. Wang, X. Geng, C. Leckie, and R. Kotagiri, "Moving shape dynamics: A signal processing perspective," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[140] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 25–34.

[141] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transportation Research Part C: Emerging Technologies*, vol. 8, no. 1, pp. 91–108, 2000.

[142] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 276–281, 2004.

[143] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, "Destination prediction by sub-trajectory synthesis and privacy protection against such prediction," in *IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 2013, pp. 254–265.

[144] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *Proceedings of the 34th international ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2011, pp. 325–334.

[145] Y. Yu and X. Chen, "A survey of point-of-interest recommendation in location-based social networks," in *Workshops at the 29th AAAI Conference on Artificial Intelligence*, 2015.

[146] J. Yuan, Y. Zheng, and X. Xie, "Discovering regions of different functions in a city using human mobility and pois," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, pp. 186–194.

[147] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 316–324.

[148] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2010, pp. 99–108.

[149] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An interactive-voting based map matching algorithm," in *Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*. IEEE Computer Society, 2010, pp. 43–52.

[150] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to find my next passenger," in *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 2011, pp. 109–118.

[151] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, "Time-aware point-of-interest recommendation," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2013, pp. 363–372.

[152] Q. Yuan, G. Cong, and A. Sun, "Graph-based point-of-interest recommendation with geographical and temporal influences," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 659–668.

[153] R. B. Zajonc, *Social facilitation*. Research Center for Group Dynamics, Institute for Social Research, University of Michigan, 1965.

[154] X. Zhan, S. Hasan, S. V. Ukkusuri, and C. Kamga, "Urban link travel time estimation using large-scale taxi data with partial information," *Transportation Research Part C: Emerging Technologies*, vol. 33, pp. 37–49, 2013.

[155] C. Zhang and S. Zhang, *Association rule mining: models and algorithms*. Springer-Verlag, 2002.

[156] C. Zhang, H. Liang, K. Wang, and J. Sun, "Personalized trip recommendation with poi availability and uncertain traveling time," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*. ACM, 2015, pp. 911–920.

[157] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "ibat: detecting anomalous taxi trajectories from gps traces," in *Proceedings of the 13th International Conference on Ubiquitous Computing*. ACM, 2011, pp. 99–108.

[158] S. Zhang, H. Hu, and H. Zhou, "An interactive internet-based system for tracking upper limb motion in home-based rehabilitation," *Medical & Biological Engineering & Computing*, vol. 46, no. 3, pp. 241–249, 2008.

[159] W. Zhang, G. Qi, G. Pan, H. Lu, S. Li, and Z. Wu, "City-scale social event detection and evaluation with taxi traces," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, p. 40, 2015.

[160] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 159–170, 2010.

[161] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, p. 29, 2015.

[162] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, p. 38, 2014.

[163] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, "Urban computing with taxicabs," in *Proceedings of the 13th International Conference on Ubiquitous Computing*. ACM, 2011, pp. 89–98.

[164] Y. Zheng and X. Xie, "Learning travel recommendations from user-generated gps traces," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 1, p. 2, 2011.